



CYBERNET[®] SERVICES

XEDIT
Extended Text Editor

User Information Manual



XEDIT COMMAND SUMMARY

<u>Command</u>	<u>Page</u>	<u>Command</u>	<u>Page</u>
ADD	2-16	NEXT	2-13
ADDLN	2-39	NOBELLS	2-52
ADDLNS	2-39	OCTCHANGE	2-45
BOTTOM	2-14	PRINT	2-15
BRIEF	2-6	Q NOS command	D-1
CHANGE	2-17	QMOD	2-23
CHANGES	2-17	QUIT	2-66
COPY	2-47	READ	2-50
COPYD	2-47	READP	2-50
Ⓢ	2-34	REPLACE	2-32
DBADL	2-42	REPLACELN	2-41
DEFTAB	2-60	RESTORE	2-53
DELETE	2-29	RMARGIN	2-62
DELETELN	2-40	STOP	2-66
DELIMIT	2-56	TABS	2-61
DEOF	2-43	TEOF	2-53
DEOR	2-42	TEOR	2-53
DLBLANKS	2-44	TOP	2-14
eEDIT	2-36	TOPNULL	2-37
END	2-66	TRIM	2-25
EXPLAIN	2-52	TRUNCATE	2-63
FBADL	2-15	VERIFY	2-6
FILE	2-64	WEOF	2-43
FINDLL	2-63	WEOR	2-43
INPUTe	2-36	WHERE	2-54
INSERT	2-33	WMARGIN	2-26
INSERTB	2-34	XEDIT NOS command	2-1
HELP	2-52	Y	2-57
linenumber	2-10	YQMOD	2-23
LISTAB	2-61	Z	2-57
LOCATE	2-10	.	2-55
MODIFY	2-21	-	2-55



CYBERNET[®] SERVICES

XEDIT
Extended Text Editor

User Information Manual


CONTROL
DATA

LIST OF EFFECTIVE PAGES

New features, as well as changes, deletions, and additions to information in this manual, are indicated by bars in the margins or by a dot near the page number if the entire page is affected. A bar by the page number indicates pagination rather than content has changed.

PAGE	REV	PAGE	REV	PAGE	REV	PAGE	REV	PAGE	REV
Cover	-								
Inside Front									
Cover	C								
Title Page	-								
ii	C								
iii/iv	C								
v/vi	C								
vii	C								
1-1 thru 1-20	C								
2-1 thru 2-69	C								
Appendixes									
A-1 thru A-6	C								
B-1	C								
C-1 thru C-5	C								
D-1 thru D-3	C								
Index-1 thru									
Index-5	C								
Comment Sheet	C								
Mailer	-								
Back Cover	-								

PREFACE

XEDIT, an extended interactive text editor, developed by the University of Minnesota, is available for use on CDC[®] NOS systems which function under CDC CYBERNET[®] Service. As a CYBERNET utility, XEDIT provides an alternative to the conventional NOS text editor (that is, EDIT). While both XEDIT and EDIT are offered by CYBERNET Interactive Service and essentially perform the same editing functions, XEDIT is considered an extended editor as it offers a series of additional, key features.

This manual presents instructional information which should help users understand how to employ XEDIT. To accommodate this instructional goal, the manual is organized in the following manner:

- Section 1 presents a condensed version of XEDIT user information. It is particularly slanted toward users who are already familiar with XEDIT, those who can learn from a cursory explanation of the editor's operation, and people who only need to employ the editor's most basic features.
- Section 2 provides a comprehensive version of XEDIT user information. Since it contains detailed instructions, it applies to readers who are interested in taking advantage of various editor options (for example, as offered through specific XEDIT command parameters) and those who want in-depth information about the editor.

While this manual is written to help new XEDIT users, it assumes the reader has a basic familiarity with NOS (especially regarding NOS file concepts). If the user needs to learn more about NOS, the CYBERNET Interactive Service Time-Sharing Tutorial (Publication No. 84000028) provides appropriate background information.

In addition, other Control Data manuals which relate to the operation of CYBERNET/NOS are:

<u>Title</u>	<u>Publication Number</u>
CYBERNET Interactive Service APL Reference Manual	84000031
CYBERNET Interactive Service Compiler Subroutine and Function Manual	84002000
CYBERNET Interactive Service FTNTS Reference Manual	84000027
CYBERNET Interactive Service CYBERLIB Library Programs Reference Manual	86605900
CYBERNET Interactive Service Project Control Guide	76073200
CYBERNET Interactive Service Time-Sharing FORTRAN Reference Manual	84001700
CYBERNET Interactive Service Time-Sharing Tutorial	84000028
CYBERNET Interactive Service Time-Sharing Usage Reference Manual	84000029
CYBERNET Service ALGOL 4 Reference Manual	84000001
CYBERNET Service BASIC Reference Manual	84000026
CYBERNET Service COBOL 4 Reference Manual	84000002
CYBERNET Service COMPASS 3 Reference Manual	84000003

<u>Title</u>	<u>Publication Number</u>
CYBERNET Service CYBER Record Manager Reference Manual	84000004
CYBERNET Service CYBERLINK User Guide	84000120
CYBERNET Service FORTRAN Extended 4 Reference Manual	84000009
CYBERNET Service Loader Reference Manual	84000014
CYBERNET Service SORT/MERGE 4 Reference Manual	84000015
CYBERNET Service UPDATE Reference Manual	84000016
MODIFY File Editing System Reference Manual	60281700
NOS 1.0 Reference Manual, Volume 1	60435400
NOS 1.0 Text Editor Reference Manual	60436100

CONTENTS

1. USING XEDIT: QUICK REFERENCE	1-1	String Editing	2-16
Overview	1-1	String Search Control	2-25
Conventions for Employing XEDIT	1-2	Line Editing	2-29
XEDIT Commands	1-4	Editing Line Numbers	2-38
Selective Command Samples	1-4	Performing Miscellaneous Editing	2-41
		Manipulating Files	2-47
		Generalized Commands	2-51
2. USING XEDIT: COMPREHENSIVE INFORMATION	2-1	Submitting Multiple Entries in a Single Line	2-55
Calling XEDIT	2-1	Tab Control	2-60
General XEDIT Conventions	2-4	Margin and Truncation Control	2-62
Positioning the File Pointer	2-8	Terminating XEDIT Execution	2-64

APPENDIXES

A XEDIT Diagnostics and Messages	A-1	C XEDIT Batch Command Processing	C-1
B Editing Direct Access Files	B-1	D In-Line Editor Usage	D-1

INDEX

FIGURES

1-1 Use of HELP Command	1-4	1-9 Use of Z and Y Commands	1-19
1-2 Use of Pointer Movement Commands	1-11	1-10 Terminating XEDIT Execution	1-20
1-3 Use of String Editing Commands	1-12	B-1 Editing Direct Access Files	B-1
1-4 Use of String Search Control Commands	1-13	C-1 Use of XEDIT Batch Processing Parameters To Create a New File	C-4
1-5 Use of Tab Control Commands	1-14	C-2 Use of XEDIT Batch Processing Parameters To Edit a File	C-5
1-6 Use of Margin Control Commands	1-15		
1-7 Use of Line Editing Commands	1-16		
1-8 Use of General XEDIT Commands	1-17		

TABLES

1-1 XEDIT Conventions	1-3	2-2 Display Code Conventions	2-46
1-2 XEDIT Commands	1-5	A-1 XEDIT Diagnostics and Messages	A-1
2-1 MODIFY Directives	2-22		

OVERVIEW

XEDIT FUNCTION

As an extended text editor, XEDIT enables NOS users to modify files by issuing various editing commands. Accordingly, the following kinds of files can be manipulated by XEDIT:

- Primary files
- Secondary files
- Indirect access files
- Direct access files (appendix B)
- Multi-file, multi-record files (that is, files containing more than one end-of-file or end-of-record mark)
- Files containing either programs, data or text (listable files)
- Files prepared in ASCII mode (the user issues the NOS ASCII command)

The user should not use XEDIT on files that do not contain recognizable text (that is, nonlistable files or nonzero byte-terminated files).

XEDIT FEATURES AND BENEFITS

The following features and benefits illustrate why XEDIT is considered an enhanced editing system:

<u>Feature</u>	<u>Benefit</u>
1. Simple command formats	XEDIT requires fewer command delimiters than certain conventional editors. As a result, users have fewer rules to remember and are less likely to commit syntax errors.
2. Multiple commands in a single line	XEDIT users can submit more than one command in a single line. Consequently, XEDIT users have greater flexibility and can speed-up their interactive editing sessions.
3. Verification of user entries	XEDIT automatically verifies that key user commands have been executed by listing those portions of the file that the user modified. Accordingly, XEDIT users save time since otherwise they would have to issue separate PRINT commands if they wanted to obtain this result.
4. Internal interrupt processing	Under XEDIT, users stay under the control of the editor even though they disrupt processing (for example, interrupting printer listings). They are not transferred back to operating system control.

<u>Feature</u>	<u>Benefit</u>
5. Editing on the basis of line numbers	While XEDIT can edit on the basis of searching for key alpha-numeric phrases (context editing) it can also unambiguously search on the basis of just a line number (line number editing).
6. Availability of permanent file commands	XEDIT users can issue certain permanent file commands while remaining under control of the editor. This feature saves time since XEDIT users do not have to terminate editor control, issue a NOS file command, and then recall the editor to perform more editing.
7. Easier line modification	By issuing the MODIFY command, XEDIT users can make changes in a line using a visual character-by-character alignment, instead of the context editing method.
8. Editing multi-file, multi-record files	By employing XEDIT, users can edit multi-file or multi-record files.
9. Tab control capability	XEDIT users can enter data more rapidly and accurately in cases where paragraph, columnar, or tabular data is being entered into a file.
10. Window capability	XEDIT users can focus the scope of all string search commands to within specific columns. Thus, the user does not need to be concerned about accidentally making a change to part of the file outside of the window area (see WMARGIN command).
11. Batch processing	In addition to normal interactive processing, XEDIT can also be called in a batch or remote job entry environment so that editing input directives can be retained for future use (see appendix C on XEDIT Batch Command Processing).
12. In-line editing mode	In addition to normal interactive processing, XEDIT can also be used to make quick "spot" changes to a file since the user can enter all commands on the same line as the editor call itself (see appendix D).

CONVENTIONS FOR EMPLOYING XEDIT

Table 1-1 presents the basic conventions which apply to XEDIT. Users can acquire more detailed information about these condensed explanations by reviewing the section 2 references listed in this table.

TABLE 1-1. XEDIT CONVENTIONS

Convention	Explanation	Section 2 Reference	Convention	Explanation	Section 2 Reference		
Issuing XEDIT commands	XEDIT sends a double question mark (??) when it expects the user to issue an editing command.	2-4	Line size	XEDIT can process file lines ranging in size from 0 to 160 characters in length. Lines greater than 160 characters will be truncated to 160 characters.	2-3		
Entering editing data	XEDIT sends a single question mark (?) after the user issues certain commands. Accordingly, the user should type input to specify exactly what should be placed in the file.	2-5			String delimiters	Within XEDIT commands, strings can be delimited by any character (except a blank, space, comma, numbers, or asterisk) not found in the delimited string.	2-6
Interrupting XEDIT processing	User can interrupt XEDIT print processing by pressing their terminal's BREAK key. XEDIT responds by waiting for the user to issue a new XEDIT command. The file pointer remains positioned at the last line in the user's file which XEDIT processed. Users can interrupt XEDIT when it is expecting the user to enter data (that is, in response to a single question mark), by pressing just carriage return.	2-7					Command parameters <u>n</u> <u>m</u> <u>ln</u>
Command syntax	After XEDIT issues a double question mark (??), enter a command in the following syntax: <u>prefix</u> <u>ln</u> <u>command</u> <u>prefix</u> = / advance one line ↑ go to top of file x suppress verification + data is on command line <u>ln</u> = line number <u>command</u> = commands in table 1-2 Extra spaces are permitted between command parameters.	2-4	File pointer movement	During its execution, XEDIT positions a file pointer at the line (in the user's file) that is currently being processed. Initially, the pointer is positioned to the first line in the file. After the user issues a command, XEDIT advances the pointer accordingly to complete the execution of that command. Then, the next command which is issued will be processed from the new pointer position. As a general rule, if the user is processing in XEDIT verify mode, the pointer is positioned at the last line which is displayed.	2-8		
Verifying XEDIT operations	By default, XEDIT prints all lines that have been affected by the issuance of an XEDIT command (that is, verify mode is in effect). BRIEF and VERIFY commands can be issued to alter verification procedures. Additionally, users can choose whether to verify the actions of individual commands. Users should enter an X to prefix a command when they want that command to be treated differently than the mode (that is, verify mode or brief mode) that is in effect.	2-6	Altering a permanent file	All XEDIT operations are performed on working copy of the edited file. Users must issue an appropriate terminating command with appropriate disposition option in order to save these operations permanently.	2-66		

†See individual commands.

[†]See individual commands.

XEDIT COMMANDS

Table 1-2 lists all the commands which can be issued under XEDIT, briefly explains their function, and specifies the portion of section 2 where more detailed information can be obtained about the command.

SELECTIVE COMMAND SAMPLES

The following sample interactive sessions demonstrate many of the most basic XEDIT commands. To facilitate easier understanding, a collection of similar commands are included in the same sample session. As a result, the following categorization applies:

- Figure 1-1 -- acquiring HELP instructions
- Figure 1-2 -- pointer movement commands
- Figure 1-3 -- string editing commands
- Figure 1-4 -- string search control commands
- Figure 1-5 -- tab control commands
- Figure 1-6 -- margin control commands
- Figure 1-7 -- line editing commands
- Figure 1-8 -- general XEDIT commands (VERIFY, BRIEF, and FILE)
- Figure 1-9 -- issuing multiple commands in a single line
- Figure 1-10 -- terminating XEDIT execution

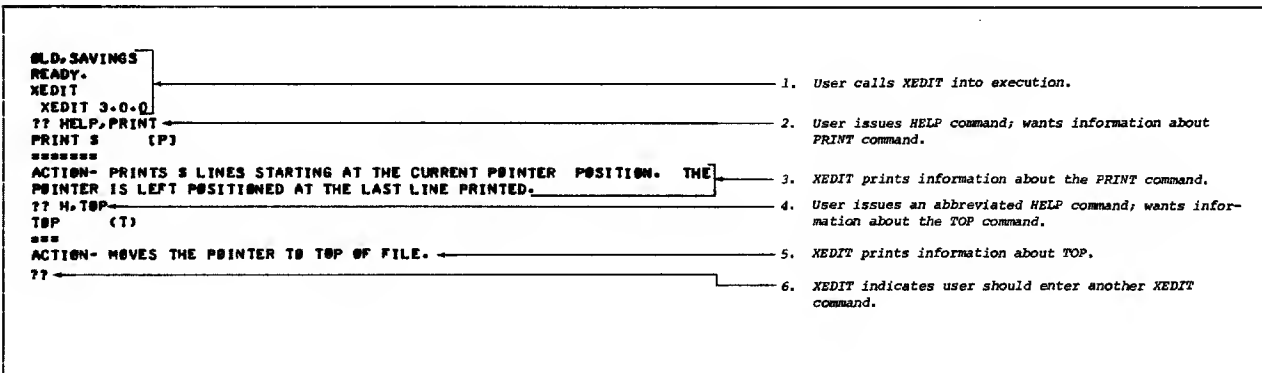


Figure 1-1. Use of HELP Command

TABLE 1-2. XEDIT COMMANDS

Command	Function	Command	Function
<p>Calling XEDIT</p> <p>OLD, <u>pfn</u> XEDIT</p> <p>NEW, <u>lfn</u> XEDIT</p> <p>GET, <u>sfn</u> XEDIT, <u>sfn</u></p> <p>ATTACH, <u>sfn</u>/M=W XEDIT, <u>sfn</u></p>	<p>The following variations apply when a user wants to call XEDIT into execution (page 2-1).</p> <p>Calls XEDIT into execution to edit an indirect access primary file which resides in a user's permanent file catalog (page 2-2).</p> <p>Calls XEDIT to edit a new empty primary file (page 2-2).</p> <p>Calls XEDIT into execution to edit an indirect access secondary file (page 2-3).</p> <p>Calls XEDIT into execution to edit a direct access file (page 2-3; see also appendix B).</p>	<p>Postfix Characters</p> <p>A</p> <p>W</p>	<p>Any command which does a string search such as CHANGE and LOCATE may be postfixed by the following postfix characters provided that a window has been defined by issuing a WMARGIN command (page 2-26).</p> <p>Forces only the first (leftmost) character of the first string to reside in the window. All other characters can reside outside of the window. Otherwise the string is not found (page 2-27).</p> <p>Forces all characters in the string to reside in the window. Otherwise the string is not found (page 2-27).</p>
<p>Interrupting XEDIT Printing</p> <p>BREAK key</p>	<p>Causes XEDIT to terminate output currently being printed at the user's terminal. Subsequently, XEDIT requests that the user enter a new command, and its file pointer is positioned at (or one line after) the last line processed (page 2-7).</p>	<p>Pointer Movement</p> <p>BOTTOM or B</p> <p>FBADL <u>n</u> or FBL <u>n</u></p>	<p>The following commands can be issued to reposition the XEDIT file pointer (page 2-9).</p> <p>Moves pointer to last line of the current record in the file (page 2-14).</p> <p>Searches for a specified number of "bad" lines. A "bad" line is defined as one which does not begin with a line number.</p>
<p>Prefix Characters</p> <p>/</p> <p>X</p> <p>↑ or A</p> <p>+</p>	<p>Any command may be prefixed with any number or combination of the following prefix characters (pages 2-7, 2-9, 2-59).</p> <p>Advances the file pointer ahead one line before processing the prefixed command (page 2-9).</p> <p>When verify mode is in effect, this prefix suppresses editor verification. When brief mode is in effect, the results of the prefixed command are verified (page 2-7).</p> <p>Moves the file pointer to the top of the file before processing the prefixed command (page 2-9).</p> <p>Forces editing data in ADD, INSERT, INSERTB, MODIFY, QMOD, REPLACE, and YQMOD to exist on the same line as the command itself rather than from a single question mark (?) prompt (page 2-59).</p>	<p><u>ln</u></p> <p>LOCATE/string/<u>n</u> or L/string/<u>n</u></p> <p>L/string1...string2/<u>n</u></p>	<p>NOTE: When the verify mode is in effect (a default condition), each bad line is printed (page 2-15).</p> <p>Performs a circular search for the line that is identified by a specified line number (ln) (page 2-10). See also "General XEDIT Conventions" (page 2-4).</p> <p>Locates <u>n</u> lines that contain a particular string of characters. In verify mode, each line containing the string is printed (page 2-11).</p> <p>Locates <u>n</u> lines that contain two specified strings of characters which are separated by an indeterminate number of other characters or phrases and string1 is followed by string2 (page 2-11).</p>

TABLE 1-2. XEDIT COMMANDS (Cont'd)

Command	Function	Command	Function
L/ <u>string1</u> --- <u>string2</u> / <u>n</u>	Locates <u>n</u> lines that contain <u>string1</u> and not followed by <u>string2</u> (page 2-12).	MODIFY or M	Allows the user to modify a particular portion of the current line. The editor first prints the line. Then, the user types his modification directly under the specific printed portion of the line that he wants to change. (See section 2 for a list of valid MODIFY directives.) XEDIT then verifies the modification by typing the affected line as it now appears in the file (page 2-21).
L/--- <u>string2</u> / <u>n</u>	Locates <u>n</u> lines which do not contain <u>string2</u> (page 2-12).		
NEXT <u>n</u> or N <u>n</u>	Advances the file pointer a specific number of lines from its current position (page 2-13).		
NEXT- <u>n</u> or N- <u>n</u>	Moves the file pointer backwards (that is, toward the top of the file) a specific number of lines (page 2-13).	QMOD <u>n</u> or QM <u>n</u>	Allows the user to modify a particular portion of one or more lines as the editor first prints a set of column numbers. Then, the user types his modification in the respective columns where he wants the modification to appear. XEDIT verifies this operation by printing the affected line(s) (page 2-23).
PRINT <u>n</u> or P <u>n</u>	Prints a specific number of lines, starting at the current pointer position (page 2-15).		
TOP or T	Moves the pointer to the top of the file (page 2-14).		
String Editing	The following commands can be issued to edit specific strings of characters which appear within a file line (page 2-16).	YQMOD <u>n</u> or YQM <u>n</u>	Allows the user to perform the same kind of modification as QMOD except that the act of column numbers is not printed (page 2-24).
ADD <u>n</u> or A <u>n</u>	Adds a specified string of input to the end of one (or more) existing lines (page 2-16).	String Search Controls	
CHANGE/ <u>string1</u> / <u>string2</u> / <u>n</u> or C/ <u>string1</u> / <u>string2</u> / <u>n</u>	Replaces every occurrence on <u>n</u> lines of one specific string of characters (that is, <u>string1</u>) with another set of specific characters (that is, <u>string2</u>) (page 2-17).	TRIM or TRIM+ or TRIM-	Toggles or sets either on or off the TRIM switch setting. If the TRIM setting is on, all commands using <u>string</u> searches (for example, LOCATE, CHANGE, etc.) ignores trailing blanks in a line (page 2-25).
C/ <u>string1a</u> ... <u>string1b</u> / <u>string2</u> / <u>n</u>	Replaces every occurrence on <u>n</u> lines of two strings (<u>string1a</u> and <u>string1b</u>) separated by an indeterminate number of other characters with another string (<u>string2</u>) (page 2-17).	WMARGIN <u>col1</u> <u>col2</u> or WM <u>col1</u> <u>col2</u>	Sets the left and right window margin column settings to define a "window" that restricts the scope of <u>string</u> searches to just the specified columns inclusively. The "W" and "A" command postfix characters are used on string search commands (such as LOCATE and CHANGE) to tell XEDIT when to put the window restriction into effect (see table 1-2 and page 2-26).
CHANGES/ <u>string1</u> / <u>string2</u> / <u>m</u> or CS/ <u>string1</u> / <u>string2</u> / <u>m</u>	Replaces <u>m</u> occurrences of <u>string1</u> with <u>string2</u> (page 2-17).		
CS/ <u>string1a</u> ... <u>string1b</u> / <u>string2</u> / <u>m</u>	Replaces <u>m</u> occurrences of two strings (<u>string1a</u> and <u>string1b</u>) separated by an indeterminate number of other characters with another string (<u>string2</u>) (page 2-17).		

TABLE 1-2. XEDIT COMMANDS (Cont'd)

Command	Function	Command	Function
Line Editing	The following commands can be issued to alter an entire line as it appears in the edited file (page 2-29).	INSERTB <u>n</u> or IB <u>n</u>	Inserts a specific number of lines into the file <u>before</u> (that is, in front of) the line designated by the current pointer position (page 2-34).
DELETE <u>n</u> or D <u>n</u>	Deletes a particular number of lines from the file, starting at the current pointer position (page 2-29).	REPLACE <u>n</u> or R <u>n</u>	Replaces a specified number of lines with another set of lines containing different entries (page 2-32).
D/ <u>string</u> / <u>n</u>	Deletes a particular number of lines from the file on the basis of specified string criteria (page 2-30).	TOPNULL or TN	Inserts a blank line at the beginning of a file and sets the file pointer to that line (page 2-37).
D/ <u>string1</u> ... <u>string2</u> / <u>n</u>	Deletes a particular number of lines that contain two specified strings, even though those strings of characters may be separated by an indeterminate number of other characters or phrases (page 2-30).	Editing Line Numbers	The following commands change file line numbers. WARNING: These commands will not modify the branch line numbers which appear in BASIC programs.
D/ <u>string1</u> --- <u>string2</u> / <u>n</u>	Deletes a particular number of lines which contain <u>string1</u> and are not followed by <u>string2</u> (page 2-29).	ADDLN <u>ln</u> <u>n</u> or ALN <u>ln</u> <u>n</u>	Adds line numbers to the entire file where none currently exist (page 2-39).
eEDIT	Used in conjunction with the INPUT command while under INPUT mode to put XEDIT back into normal command EDIT mode (double question mark (??); see page 2-36).	ADDLNS <u>ln</u> <u>n</u> or ALNS <u>ln</u> <u>n</u>	Performs same function as ADDLN except that the line numbers are followed by a space (page 2-39).
Ⓢ	By pressing the carriage return, a user can insert an <u>unspecified</u> number of lines into the file <u>after</u> the line designated by the current pointer position (page 2-35).	DELETEN or DLN	Deletes all line numbers in the file (page 2-40).
INPUT <u>e</u>	Same effect as the Ⓢ command except that the user can issue most XEDIT commands while under INPUT mode to make quick changes to the line just entered, by prefixing the command with the escape character <u>e</u> (page 2-36).	REPLACEN <u>ln</u> <u>n</u> or RLN <u>ln</u> <u>n</u>	Replaces the existing set of line numbers in a file with a new set of line numbers (page 2-41).
INSERT <u>n</u> or I <u>n</u>	Inserts a specific number of lines into the file <u>after</u> (that is, behind) the line designated by the current pointer position (page 2-33).	Miscellaneous Editing Commands	The following commands allow the user to modify the contents of his file in a variety of miscellaneous ways:
		DBADL <u>n</u> or DBL <u>n</u>	Searches for and deletes a specified number of "bad" lines (beginning at the current pointer position). A "bad" line is defined as one which does not begin with a line number (page 2-42).
		DEOF <u>m</u> or DF <u>m</u>	Deletes the next <u>m</u> number of end-of-file marks (page 2-43).

TABLE 1-2. XEDIT COMMANDS (Cont'd)

Command	Function	Command	Function
DEOR <u>m</u> or DR <u>m</u>	Deletes the next <u>m</u> number of end-of-record marks (page 2-42).	COPY <u>fname</u> /--- <u>string2</u> / <u>n</u>	Same as above except that the <u>string-line</u> count <u>n</u> is decremented only if the line does not contain <u>string2</u> (page 2-48).
DLBLANKS <u>n</u> or DLB <u>n</u>	Deletes leading blanks from <u>n</u> lines that appear in the file from the current pointer position (page 2-44).	COPYD <u>fname</u> <u>n</u> or COPYD <u>fname</u> / <u>string</u> / <u>n</u> or COPYD <u>fname</u> / <u>string1</u> ... <u>string2</u> / <u>n</u> or COPYD <u>fname</u> / <u>string1</u> --- <u>string2</u> / <u>n</u> or COPYD <u>fname</u> /--- <u>string2</u> / <u>n</u>	Performs the same function as COPY except the copied lines are deleted from the edit file (page 2-49).
OCTCHANGE <u>oct1 oct2 n</u> or OC <u>oct1 oct2 n</u>	Converts the octal display code of a specific character or string to a different octal code character or string. Typically, users employ this to acquire an executable function (for example, rubout or line feed) that they normally could not enter into their file (page 2-45).	READ <u>fname1</u> ... <u>fnamen</u>	Copies the contents of specific <u>local</u> files into the file which the user is editing (page 2-51).
WEOF or WF	Writes an end-of-file mark on the file at the position <u>before</u> the line currently designated by the file pointer (page 2-44).	READP <u>fname1</u> ... <u>fnamen</u>	Copies the contents of specific <u>permanent</u> files into the file which the user is editing (page 2-50).
WEOR or WR	Writes an end-of-record mark on the file at the position <u>before</u> the line currently designated by the file pointer (page 2-43).	General Commands	The following commands perform generalized, nonediting functions:
File Commands	The following commands let the user manipulate <u>entire</u> files:	BRIEF or BRIEF + or BRIEF - or BR	Suppresses XEDIT verification procedures (page 2-6). Turn Brief mode on.
COPY <u>fname</u> <u>n</u> or	Copies <u>n</u> lines from the edit file onto file <u>fname</u> ; the edit file remains intact (page 2-48).	EXPLAIN	Turn Brief mode off.
COPY <u>fname</u> / <u>string</u> / <u>n</u> or	Copies inclusively all lines from the edit file current pointer position to file <u>fname</u> until either the <u>string-line</u> count <u>n</u> is satisfied or END OF FILE is encountered (page 2-48).	HELP, <u>cmd</u> or H, <u>cmd</u>	Gives the user a more detailed description of the most recent error message that has been printed by XEDIT. If the message was not as a result of an error condition, no explanation is given (page 2-52).
COPY <u>fname</u> / <u>string1</u> ... <u>string2</u> / <u>n</u> or	Same as above except <u>string</u> may be specified as two strings (<u>string1</u> and <u>string2</u>) that are separated by an indeterminate number of other characters (page 2-48).	NOBELLS or NB	Requests information about a specific XEDIT command (page 2-52).
COPY <u>fname</u> / <u>string1</u> --- <u>string2</u> / <u>n</u> or	Same as above except that the <u>string-line</u> count <u>n</u> is decremented only when a line contains <u>string1</u> which is not followed by <u>string2</u> (page 2-48).		Prevents XEDIT from ringing the user's terminal bell when error messages are printed (page 2-52).

TABLE 1-2. XEDIT COMMANDS (Cont'd)



Command	Function	Command	Function
RESTORE or REST	Cancels any changes that have been made after the occurrence of any of the RESTORE conditions (page 2-53).	LISTAB or LT	Lists the current tab character and tab stop column positions (page 2-61).
TEOF or TEOF + or TEOF -	Toggles or sets either on or off the printing of the message --EOR-- (page 2-54).	TABS <u>t</u> 1... <u>t</u> 8 or TAB <u>t</u> 1... <u>t</u> 8	Defines tab stop column positions. Defaults are 11, 18 and 30 (COMPASS) (page 2-61).
TEOR or TEOR + or TEOR -	Toggles or sets either on or off the printing of the message --EOR-- (page 2-54).	Margin Control Commands	The following commands allow the user to control the right margin of the user's file (page 2-62).
VERIFY or VERIFY + or VERIFY - or V	Initiates XEDIT verification procedures. NOTE: XEDIT operates in verify mode by default (page 2-6).	FINDLL <u>n</u> or FLL <u>n</u>	Finds and lists <u>n</u> long lines, where long lines are defined to be lines having more characters than the current RMARGIN setting (page 2-65).
WHERE or W	Prints the current line count (that is, the number of lines from the top of the file to the line designated by the current pointer position) (page 2-54).	RMARGIN <u>m</u> or RM <u>m</u>	Sets the right margin character position to <u>m</u> (page 2-62).
<u>n</u>	Advances the file pointer <u>n</u> lines and reexecutes the last command that the user entered (page 2-55).	TRUNCATE <u>n</u> or TRUNC <u>n</u>	Truncates <u>n</u> long lines to RMARGIN length starting at the current pointer position (page 2-63).
<u>-n</u>	Advances the file pointer <u>n</u> lines and reexecutes the last Z or Y command that the user entered (page 2-55).	Multiple Commands	The following commands let a user enter more than one command in a single line of entry:
Tab Control Commands	The following commands allow the user to perform tabbing (column format spacing) in conjunction with the INSERT, INSERTB, REPLACE, INPUT and  input requesting commands:	DELIMIT <u>char</u> or DEL <u>char</u>	Establishes a particular character as the delimiter which the user will employ to separate multiple lines of input (page 2-56).
DEFTAB <u>char</u> or DT <u>char</u>	Establishes <u>char</u> as the tab operator for later use when inputting editing data via the INSERT, INSERTB, REPLACE and  commands (page 2-60).	Z\$ <u>cmd</u> 1\$ <u>cmd</u> 2\$...\$ <u>cmd</u> n	Allows user to enter several commands in one line of entry; XEDIT lists each command before it executes that command (page 2-57).
		Y\$ <u>cmd</u> 1\$ <u>cmd</u> 2\$...\$ <u>cmd</u> n	Performs same function as Z except that commands are not listed before being executed (page 2-57).

TABLE 1-2. XEDIT COMMANDS (Cont'd)

Command	Function	Command	Function
Terminating XEDIT	The following commands can be issued to terminate XEDIT execution:		(in accordance with the user's <u>mode</u> parameter). Subsequently, XEDIT execution is automatically resumed† (page 2-65).
END <u>fname mode</u> or E <u>fname mode</u>	Terminates text editing and allows the user to dispose of his edited file (in its modified form) by saving it as a permanent file† (page 2-67).	QUIT <u>fname mode</u> or Q <u>fname mode</u>	Performs same function as END† (page 2-67).
FILE <u>fname mode</u> or F <u>fname mode</u>	Temporarily suspends text editing and saves or replaces the edited file (in its modified form) as a permanent and/or local file	STOP	Terminated text editing without writing the modified edited file anywhere (page 2-66).

†Valid mode parameters are:

SAVE = edited file should be saved as a new indirect access permanent file
or
S

REPLACE = edited file should replace an existing indirect access permanent file
or
R

LOCAL = edited file should be written onto a local file. Default: If the user does not enter any mode parameter, or XEDIT assumes LOCAL should apply unless a direct access file is attached. LOCAL mode is illegal for direct access files.
L

COPY = edited file changes should be copied to file fname, if file fname is a direct access file attached in write mode.
or This parameter is default for direct access files (that is, the user may simply type END or FILE). If the direct access file has not been attached in write mode, the user will get an error message.
C

RL = performs both LOCAL and REPLACE specifications

SL = performs both the SAVE and LOCAL specifications

Additionally, the fname parameter should specify the name which will be applied to the edited file when it is saved, replaced, or made local. Default: If no fname parameter is entered, XEDIT assumes the file's original name should be employed. In this instance, commas must separate the command from the mode parameter. Some examples:

QUIT,,REPLACE
or
E,,RL
or
F,,RL

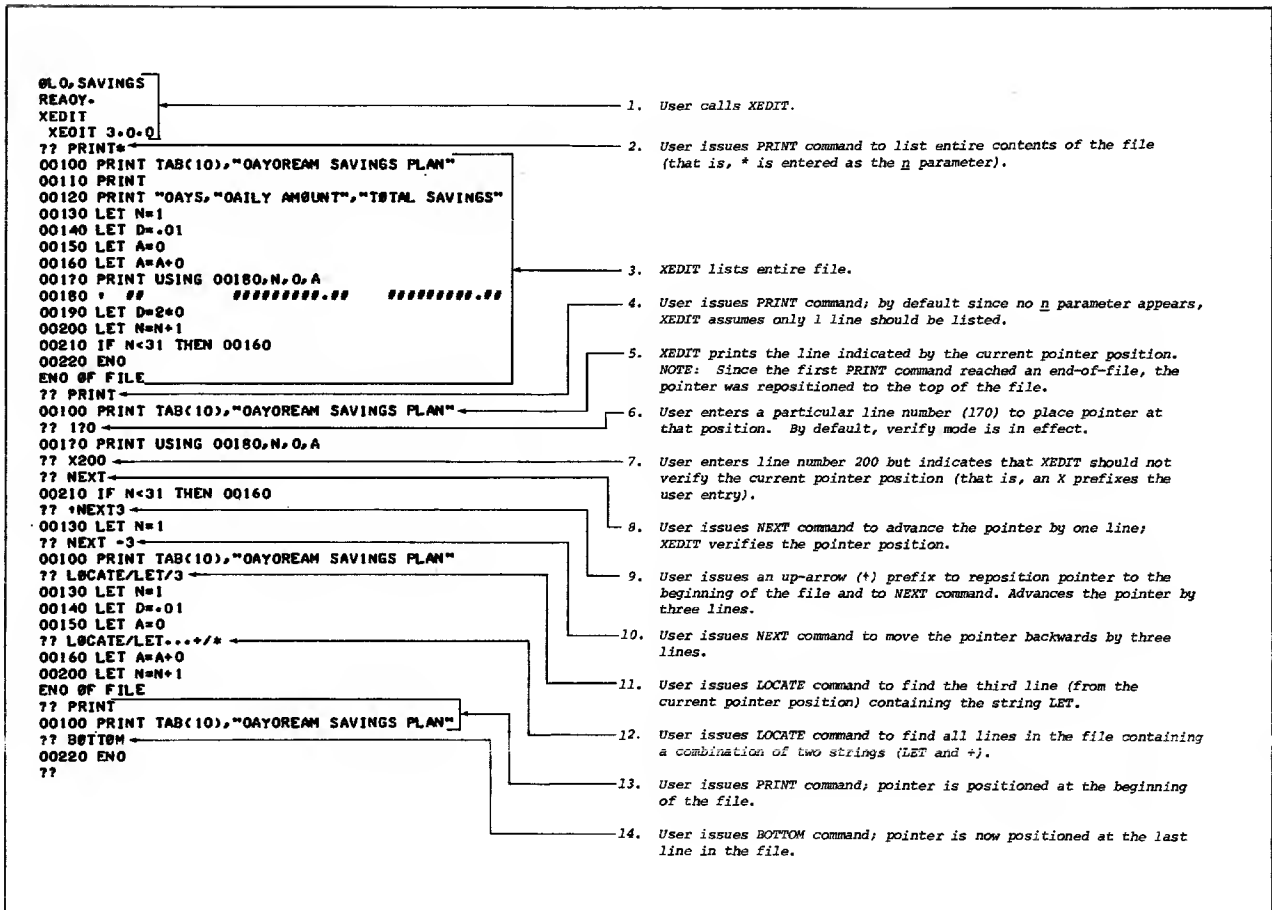


Figure 1-2. Use of Pointer Movement Commands

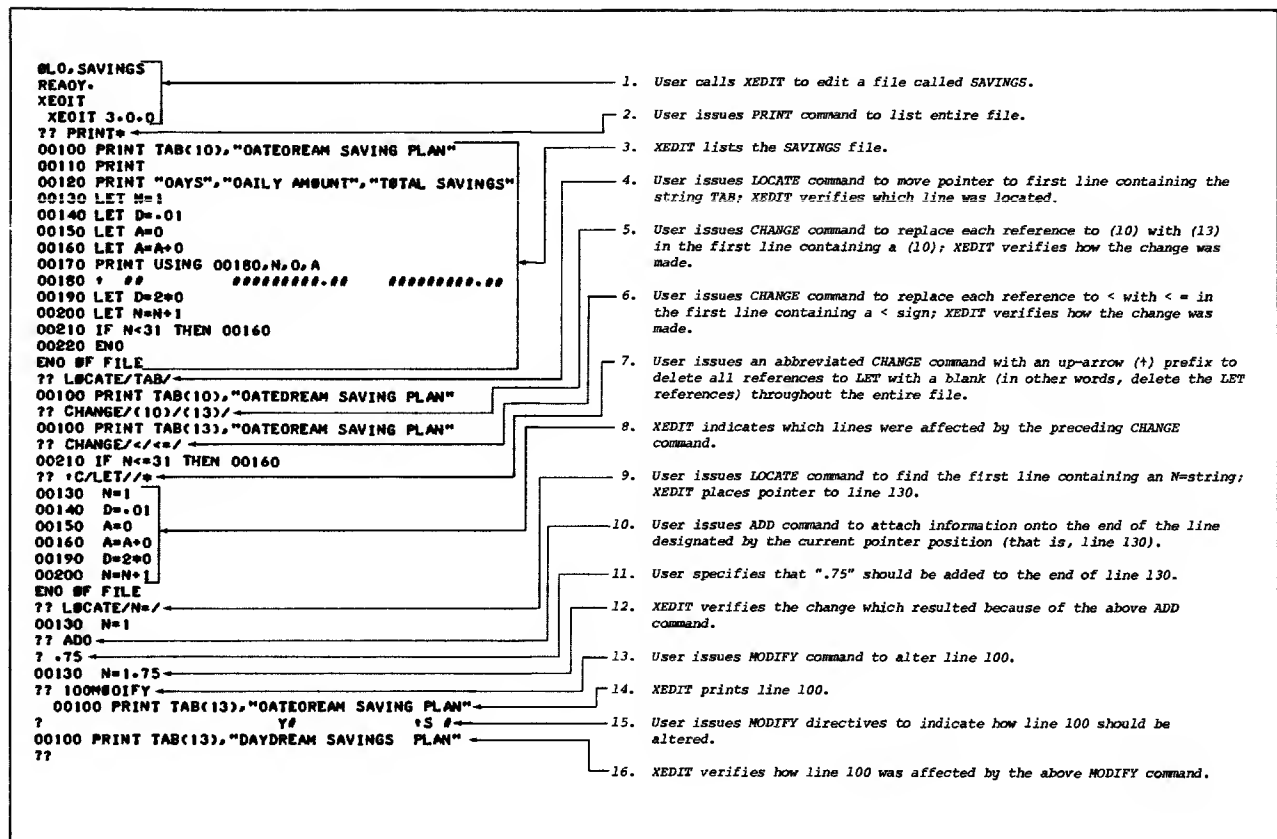


Figure 1-3. Use of String Editing Commands

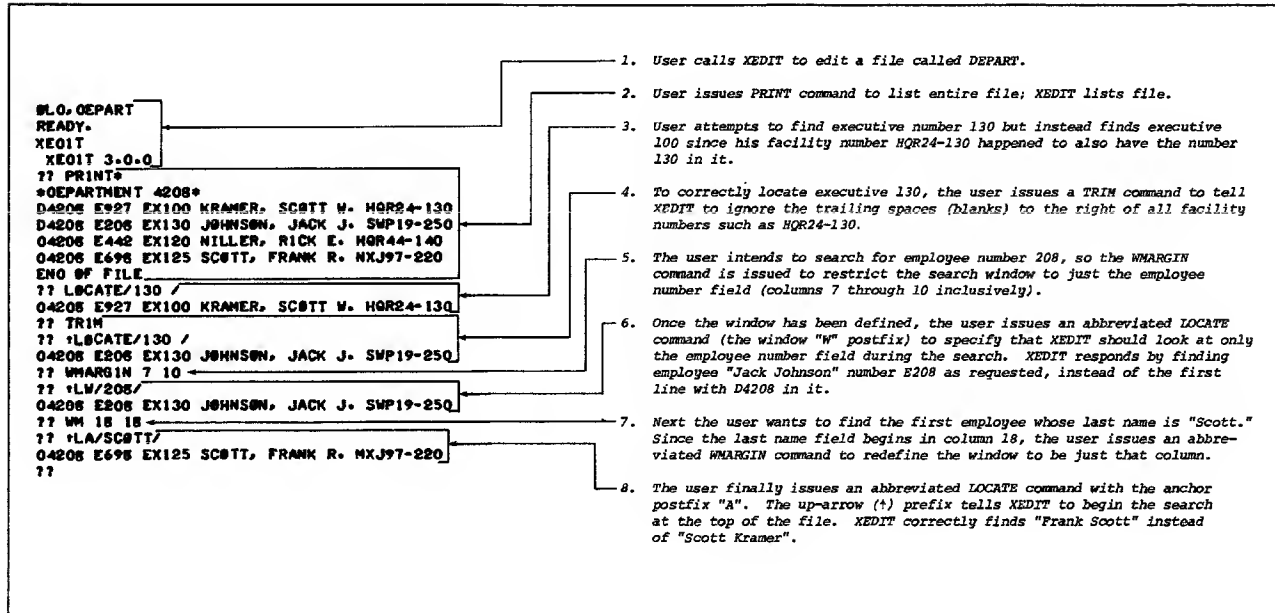


Figure 1-4. Use of String Search Control Commands

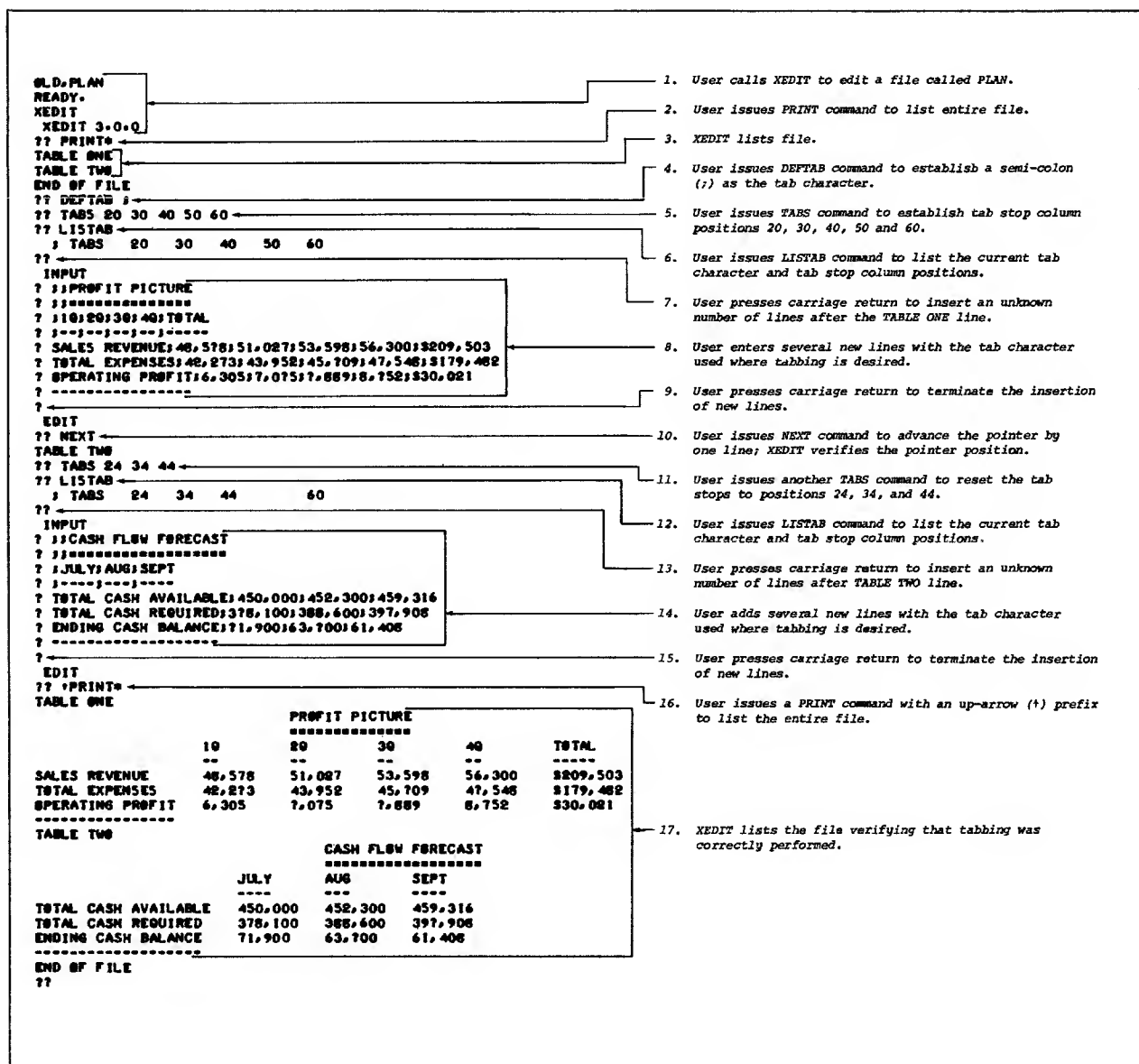


Figure 1-5. Use of Tab Control Commands

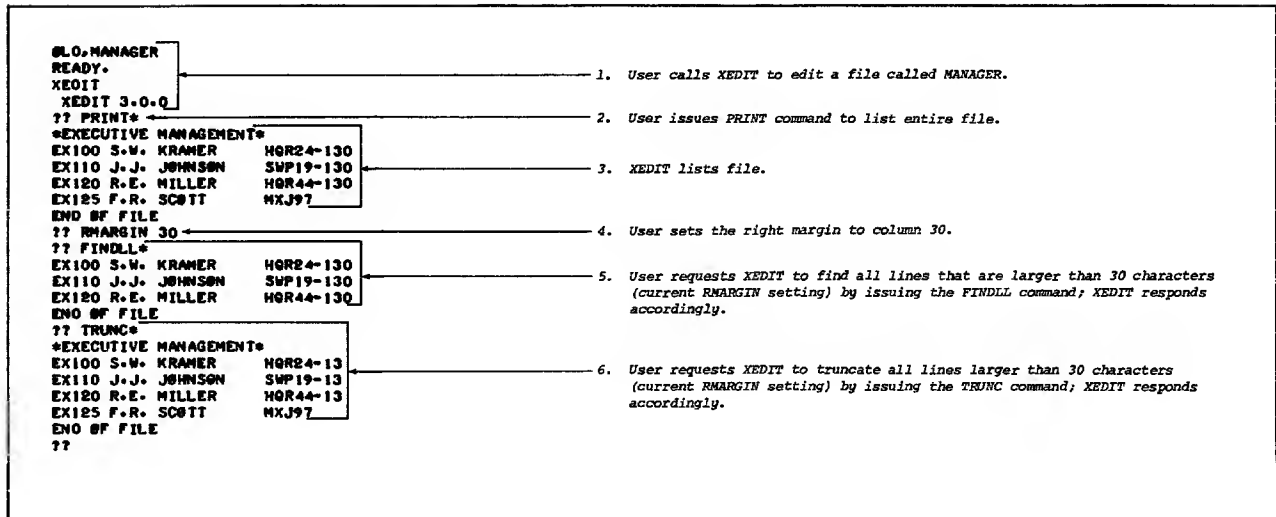


Figure 1-6. Use of Margin Control Commands

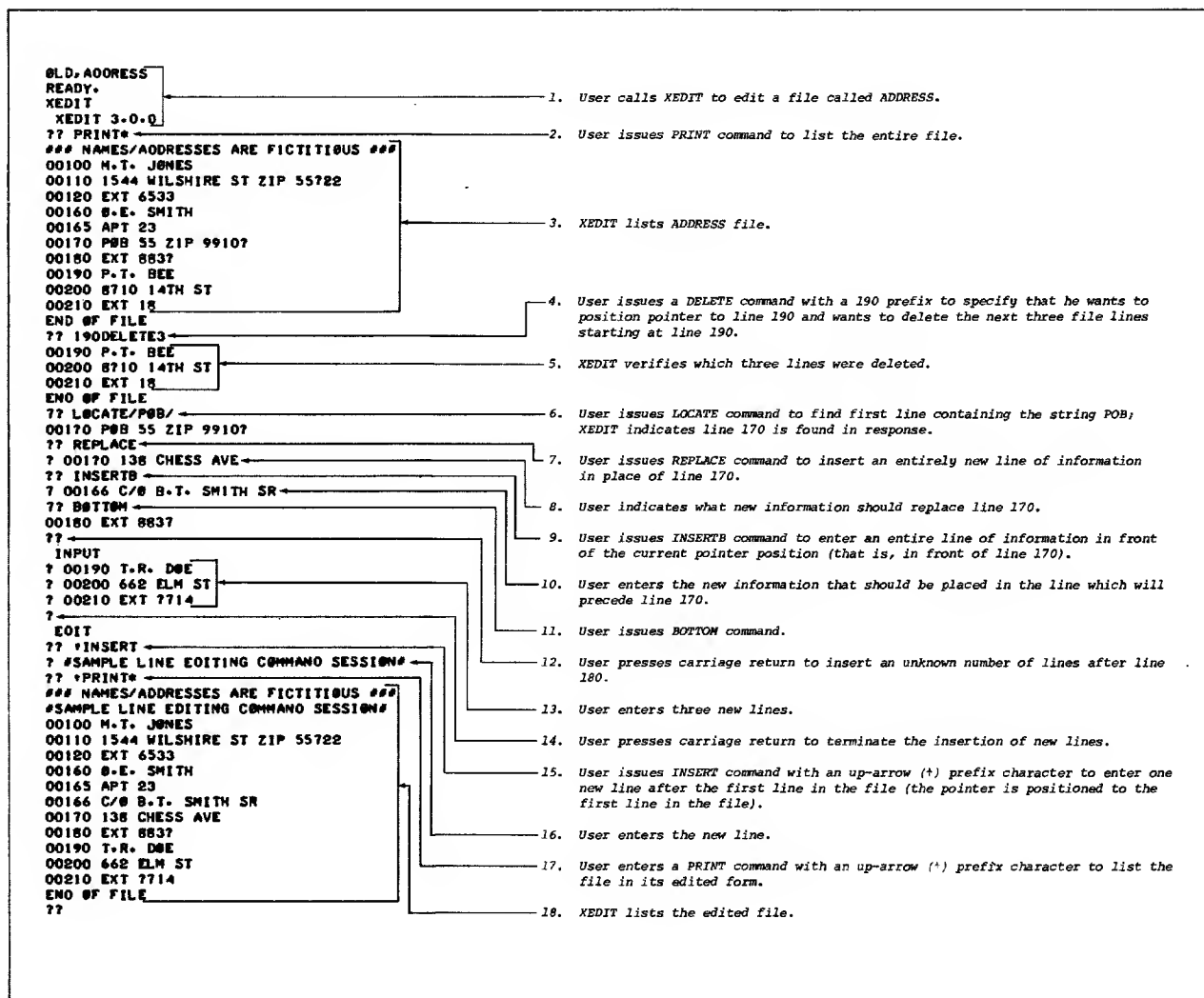


Figure 1-7. Use of Line Editing Commands

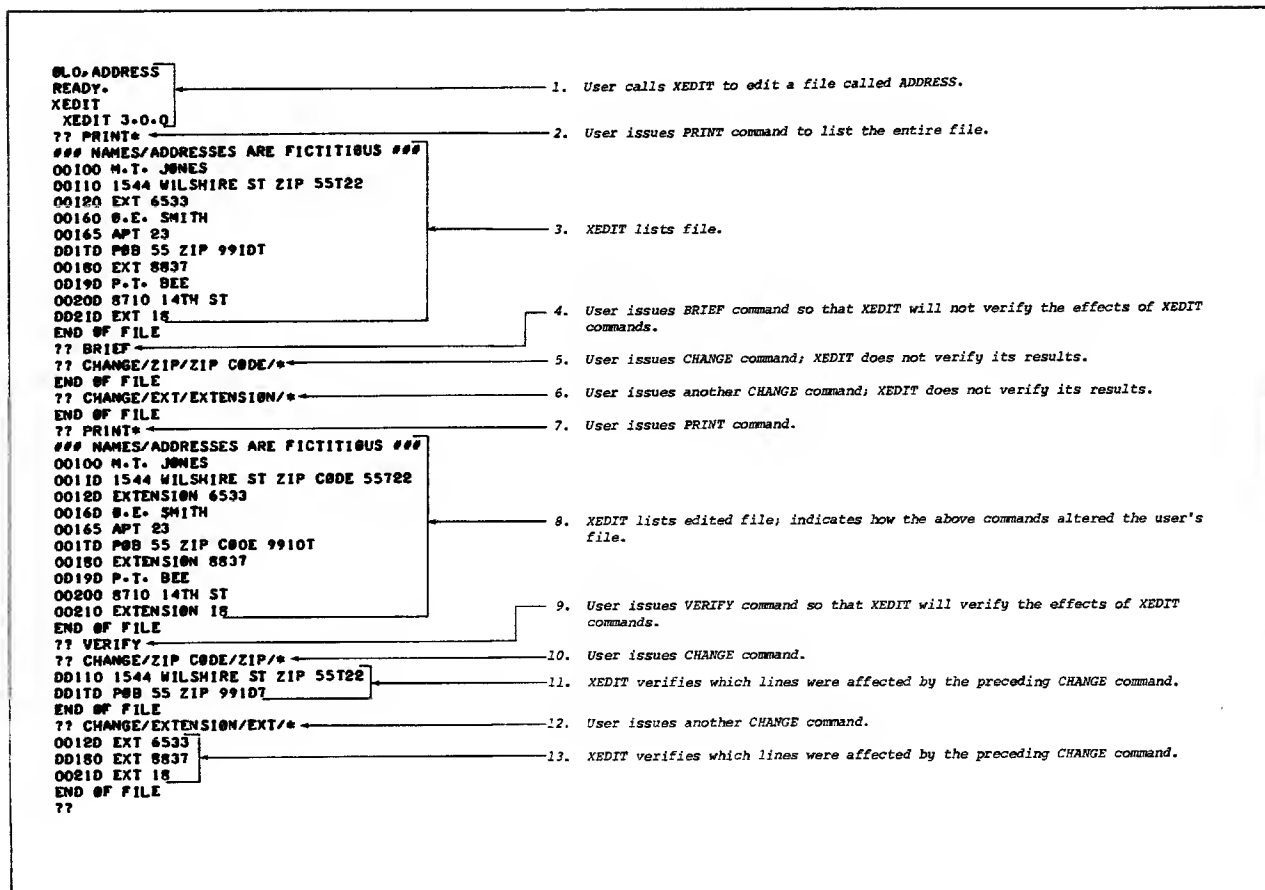


Figure 1-8 (Part 1). Use of General XEDIT Commands (VERIFY and BRIEF)

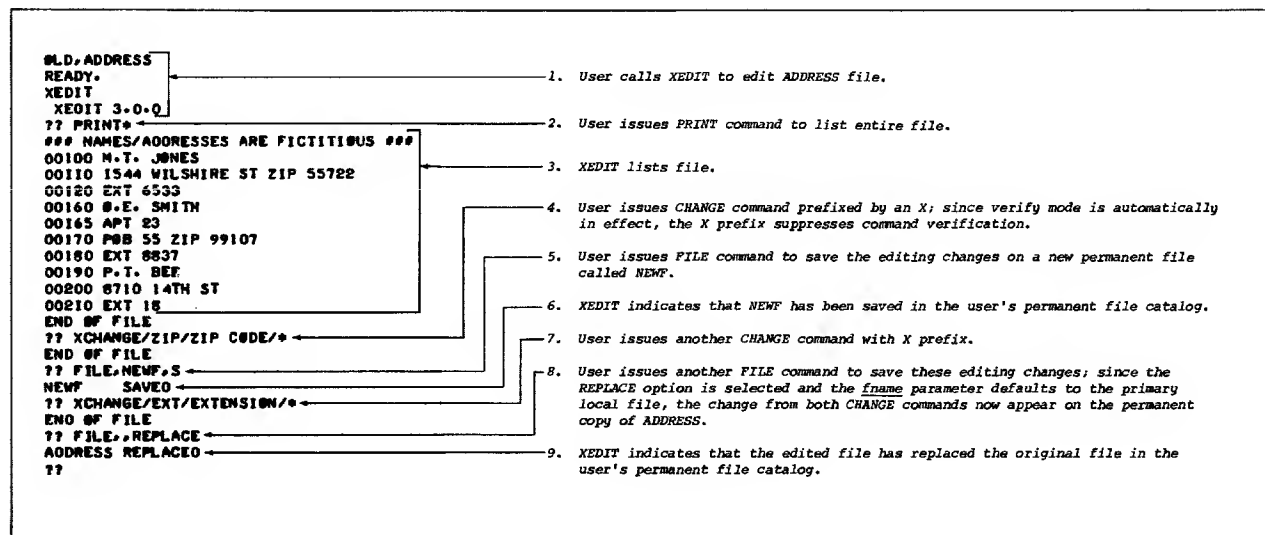


Figure 1-8 (Part 2). Use of General XEDIT Commands (FILE)

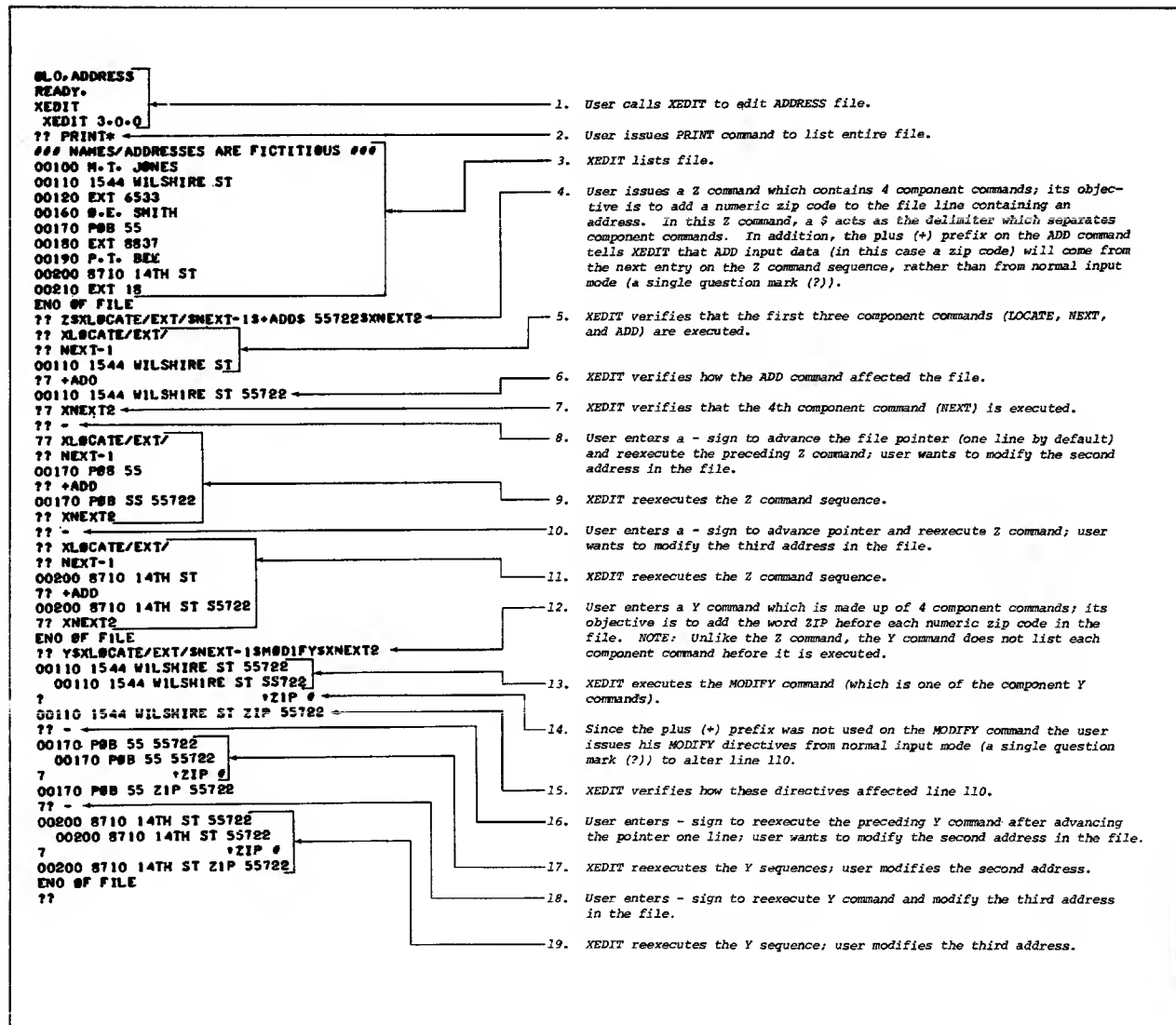


Figure 1-9. Use of Z and Y Commands and Plus (+) Prefix Character

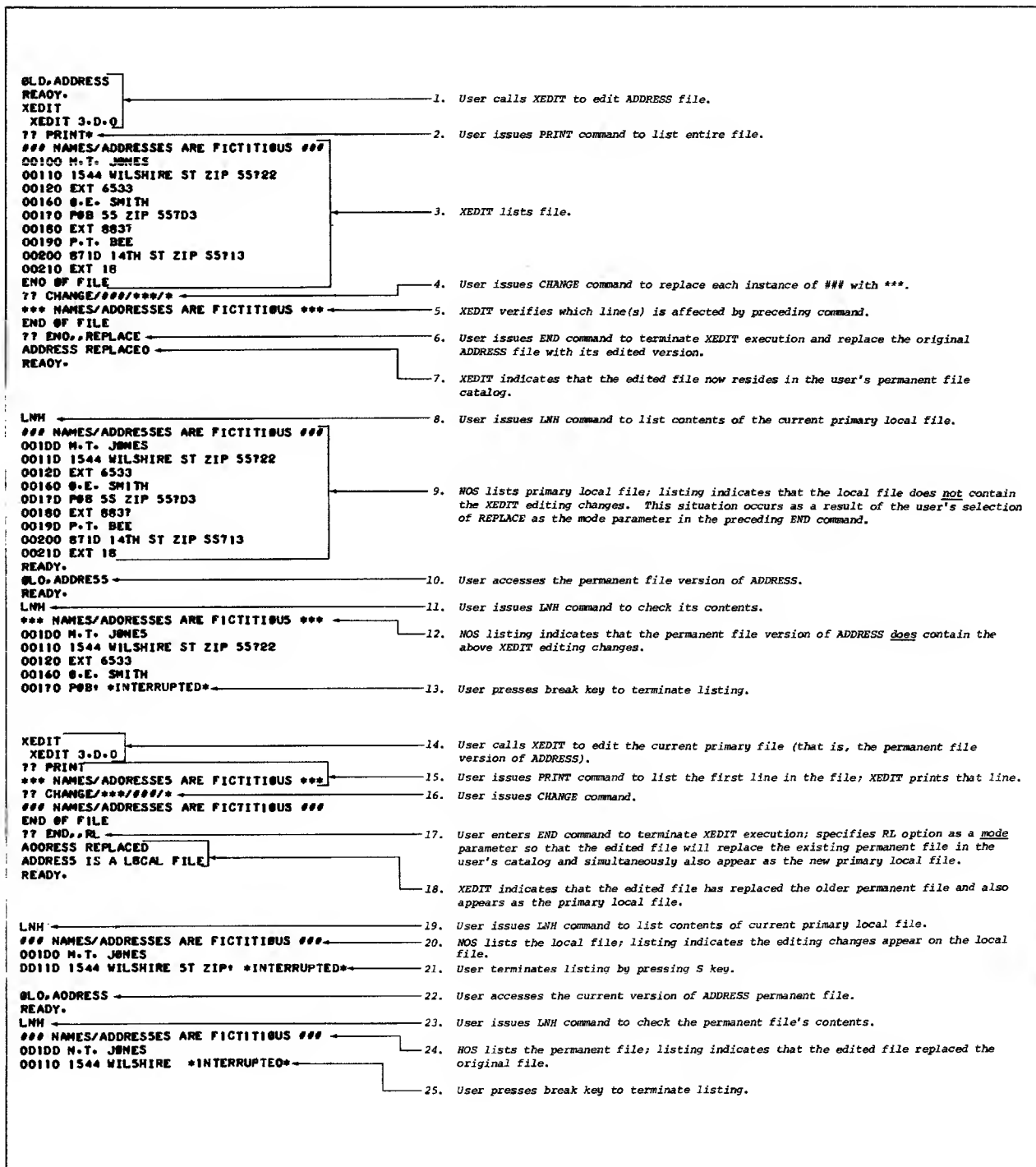


Figure 1-10. Terminating XEDIT Execution

This section discusses in detail how users should employ XEDIT to modify files by issuing editing commands. Accordingly, it is divided into the following sections:

- Calling XEDIT into execution (page 2-1)
- General XEDIT conventions (page 2-4)
- Moving the file pointer (page 2-10)
- String editing (page 2-16)
- Line editing (page 2-29)
- Editing line numbers (page 2-38)
- Miscellaneous editing (page 2-41)
- Manipulating files (page 2-47)
- Generalized commands (page 2-51)
- Submitting multiple entries (page 2-55)
- Tab control (page 2-60)
- Margin and truncation control (page 2-62)
- Terminating XEDIT execution (page 2-64)

CALLING XEDIT

The first step which users must take when employing XEDIT is to call the editor into execution.[†] However, the procedure for calling XEDIT varies depending upon whether primary or secondary files are being edited (for this discussion, direct access files will be considered secondary).

EDITING PRIMARY INDIRECT ACCESS FILES

When users want to edit indirect access files as primary files, they submit the following entries to call XEDIT into execution.

[†]Users of certain terminals should issue an NOS ROUT command before calling XEDIT (in order to insure correct terminal positioning when users employ MODIFY, QMOD, and YQMOD commands under the editor). See the CYBERNET Interactive Service Time-Sharing Tutorial listed in the Preface for a description of the ROUT command.

Old Files

OLD, pfn (CR)
XEDIT (CR)[†]

where:

pfn = name of an existing (primary) indirect access permanent file which the user wants to edit.

Example

```
OLD, ADDRESS
READY.
XEDIT
XEDIT 3.0.0
??
```

Explanation: User calls XEDIT to edit an existing permanent file called ADDRESS as the primary file.

New Files

NEW, lfn (CR)
XEDIT (CR)[†]

where:

lfn = local (primary) file name that the user wants to assign to the new file.

Example

```
NEW, BTFILE
READY.
100 INPUT N
110 IF N < 0 THEN TAG1
120 PRINT SQR (N)
130 STOP
140 PRINT "NEGATIVE NOT ALLOWED"
150 END
PACK
READY.
XEDIT
XEDIT 3.0.0
?? CHANGE/TAG1/140/*
110 IF N < 0 THEN 140
END OF FILE
?? E,,SL
BTFILE SAVED
BTFILE IS A LOCAL FILE
READY.
```

Example Explanation: User calls XEDIT to edit a newly created file called BTFILE. After entering BASIC coding, the user issues an NOS PACK command (to pack the file). Then, he calls XEDIT into execution by entering XEDIT. XEDIT sends a double question mark (??) indicating that the user can now enter an editing command. User enters a CHANGE command since he now knows what line number identifies the statement temporarily identified by TAG1. XEDIT proceeds to verify the CHANGE command's effect upon line 110. Finally, the user issues an END command to save BTFILE as both a local file and a permanent indirect access file.

[†] See also appendix D, for the method of using XEDIT for in-line editing.

EDITING SECONDARY INDIRECT ACCESS FILES

If users want to edit indirect access files as secondary files, they call XEDIT by entering:

Format

GET, sfn (CR)
XEDIT, sfn (CR)[†]

where:

sfn = name of the secondary indirect access file which the user wants to edit.

Example

OLD, BPGM
READY.
GET, BDATA
READY.
RNH
ERROR IN DATA FILE AT LINE 260
RUN COMPLETE.

XEDIT, BDATA
XEDIT 3.0.0
??

Example Explanation: User calls an existing program file (called BPGM) which takes its input data from a file called BDATA. After calling both of these files, the user issues an RNH command to run BPGM. However, during its execution BPGM encounters an error in the input data file (that is, in BDATA). Subsequently, the user calls XEDIT to edit the BDATA file (which is a secondary file) and correct the input error.

EDITING DIRECT ACCESS FILES

To call XEDIT when manipulating direct access files, users enter:

Format

ATTACH, sfn/M=W (CR)
XEDIT, sfn (CR)[†]

where:

sfn = name of the direct access file which the user wants to edit.

Example

ATTACH, PFTOP/M=W
READY.
XEDIT, PFTOP
XEDIT 3.0.0
??

Explanation: User calls XEDIT to edit a permanent direct access file called PFTOP.

For more detailed information on editing direct access files, see appendix B.

[†]See also appendix D, for the method of using XEDIT for in-line editing.

GENERAL XEDIT CONVENTIONS

Once users call XEDIT into execution, they can issue appropriate XEDIT commands to manipulate the file in question. However, before users issue any command, they should review the basic editing conventions and rules which govern XEDIT.

ISSUING XEDIT COMMANDS IN CORRECT COMMAND SYNTAX

XEDIT transmits a double question mark (??) whenever it expects the user to enter an XEDIT command.[†] Accordingly, if the user enters a valid command, XEDIT simply processes that command. A command is considered valid when the following conventions are followed:

1. The general syntax for an XEDIT command is as follows:

prefixlncommand

where:

prefix = optional prefix characters for use as indicated below:

- / to advance the pointer one line before processing the prefixed command (see page 2-9).
- † or ^ to reposition the pointer to the beginning-of-file before processing the prefixed command (see page 2-9). This character is octal 76.
- X to temporarily suppress verify or brief mode while processing the prefixed command (see page 2-7).
- + to tell XEDIT that editing data exists on the same line as the command itself when used in conjunction with the DELIMIT, Y, or Z command; the commands ADD, INSERT, INSERTB, MODIFY, QMOD, REPLACE, and YQMOD are the only commands that logically use the + prefix, thus the + prefix will be ignored when using it with other commands (see page 2-59 for examples).

Prefix characters can be in any order and combination.

ln = optional line number prefix which specifies the line at which the user wants the specified command to be executed. This prefix works identical to the find line number (ln) command with one added feature. If the desired line is not found, the command is not executed, an informative message is given, and the line at the current pointer position is printed (if in VERIFY mode).

command = any legal XEDIT command as given in table 1-2.

[†]A double question mark is not transmitted if the user has assigned the input or output file to a mass storage device or to a file using XEDIT Batch control card I= parameter (see appendix C).

2. The command must be spelled correctly.
3. Command parameters must follow the sequence shown in the documented command format.
4. Command parameters must stay within their maximum and minimum numerical limits. However, an asterisk (*) can be used in place of any n or m parameter and has the value 99999.
5. The command itself must be entered immediately at the position where the terminal stopped after XEDIT sends the double question mark.
6. No embedded blanks can appear in the command itself or between ln and command fields.
7. If the first character in the command is alphabetic, the command must be separated from its parameters by either a comma or a space.
8. Extra spaces may appear between command parameters.

If the double question mark (??) appears and the user issues any entry other than a valid command, XEDIT responds with one of the messages described in appendix A.[†]

ENTERING EDITING DATA

Certain XEDIT commands require that the user explicitly enter data to indicate how his file should be modified. In these instances, XEDIT transmits a single question mark (?) to specify that the user can now submit his particular entry. ADD, MODIFY, QMOD, YQMOD, INSERT, INSERTB and REPLACE are examples of commands that provoke this response. For instance, consider the following situation:

<u>Example</u>	<u>Explanation</u>
?? 110MODIFY	XEDIT sends double question mark; user responds by issuing a MODIFY command to alter line 110.
00110 1544 WILSHIRE ST ZIP 55722	
? 2	XEDIT prints line 110.
00110 2544 WILSHIRE ST ZIP 55722	
??	XEDIT transmits single question mark; user responds by indicating how he wants line 110 modified (that is, he supplies editing data).
	XEDIT prints the modified version of line 110.

When XEDIT expects editing data to be entered, the user can interrupt this procedure by simply entering a carriage return. XEDIT then transmits a double question mark (??) to specify that the user can now enter a new command.

Editing data can also be entered in the same line as the command by using a plus (+) prefixed command in a Y or Z command list or by entering a plus prefixed command in a delimited command sequence (see Submitting Multiple Entries in a Single Line, page 2-55).

[†]See appendix A for a list of all XEDIT diagnostic messages.

STRING DELIMITERS

Certain XEDIT commands allow the user to modify small pieces of information which appear within a file line. These alphanumeric words, phrases, or numbers are called strings. When users enter string commands, they separate strings from the command itself, other parameters and other strings by employing string delimiters. Throughout this manual, delimiters are represented by a slash (/) character, as shown in the following LOCATE command format:

LOCATE/string/n (CR)

However, under XEDIT, a delimiter can be any character (other than a space, number, asterisk, or comma) not found in the string that is being delimited. Consider the following example: A user wants to locate the first line in the file which contains the phrase (that is, string) IF N=. Accordingly, any of the following LOCATE commands are valid and identical in purpose, even though their string delimiters vary.

LOCATE/IF N = / (CR)

LOCATE"IF N = " (CR)

LOCATE ZIF N = Z (CR)

VERIFYING XEDIT OPERATIONS

VERIFY Mode vs BRIEF Mode

By default, XEDIT assumes that users will want the editor to verify the effect of an XEDIT command after the user issues that command. For example, suppose a user wants to delete two lines in the file, beginning at the current file pointer position. Accordingly, the user would issue a DELETE command. Under XEDIT, the editor automatically lists the lines which were deleted so that the user can verify that the command in fact performed the task which was intended. This automatic verification occurs when XEDIT operates in VERIFY mode.

While VERIFY mode is a standard, default condition under XEDIT, users can negate this procedure by issuing a BRIEF command so that verification does not automatically occur. In other words, if a user does not want to verify the effect of commands like DELETE, he can issue a BRIEF command and verification will be suppressed. When users suppress automatic verification by issuing a BRIEF command, they are said to operate under BRIEF mode.

Users can select whether they want verification or no verification by choosing the command from the following table:

Action	Legal Commands		
verification	VERIFY (CR)	VERIFY+ (CR)	BRIEF- (CR)
no verification	BRIEF (CR)	BRIEF+ (CR)	VERIFY- (CR)

The abbreviations for BRIEF and VERIFY take the following forms respectively:

BR (CR)

V (CR)

The following example illustrates the difference between VERIFY mode and BRIEF mode. In both instances, the user issues a CHANGE command to delete the word (that is, string) ZIP from every line in the file. In VERIFY mode, XEDIT executes the command and lists each line where the specified change was made. In BRIEF mode, the command is merely executed.

VERIFY Mode

```
?? CHANGE/ZIP/*
00110 1544 WILSHIRE ST 55722
00170 POB 55 55703
00200 8710 14TH ST 55713
END OF FILE
??
```

BRIEF Mode

```
?? CHANGE/ZIP/*
END OF FILE
??
```

Temporarily Suppressing VERIFY or BRIEF Mode

When editing under either VERIFY mode or BRIEF mode, users may periodically want to execute a single command and have it processed as if it were under the alternate mode -- without having to issue a BRIEF or VERIFY command. This can be accomplished by prefixing the deviant command with an X. A typical example (shown below) occurs when a user is operating in VERIFY mode but does not have the actions of a particular CHANGE command verified. In this instance, the user would prefix the command in the following manner:

<u>Example</u>	<u>Explanation</u>
?? VERIFY	User issues VERIFY command.
?? LOCATE/ZIP/	User issues unprefix LOCATE command.
00110 1544 WILSHIRE ST ZIP CODE 55722	XEDIT verifies effect of preceding LOCATE command.
?? XCHANGE/ZIP CODE/ZIP/	User issues a CHANGE command with an X prefix. XEDIT suppresses its verification.
??	

INTERRUPTING XEDIT PROCESSING

Users can terminate both the printing of XEDIT output and all XEDIT line or editing input requests (signified by the appearance of a single question mark).

To terminate the transmission of XEDIT terminal output, the user must press the BREAK key.

When a user interrupts output transmission, XEDIT reacts in the following manner:

- The editor stops processing the command that it was executing.
- XEDIT sends a double question mark to the user, indicating that a new XEDIT command should be issued.

- The file pointer is positioned at (or one line after) the last line that was being processed when the user interrupted XEDIT execution. Under VERIFY mode, this may mean that the pointer is positioned several lines beyond the last line that was verified. Accordingly, users are advised to issue a PRINT command immediately after interruption to determine the current pointer position.

If the interruption occurs while a Z, Y, or delimited command sequence is executing, the remaining component commands in the command list will be skipped.

To terminate XEDIT requests for editing input or line input (via commands such as INSERT, INSERTB, REPLACE, INPUT), the user simply presses the carriage return.

When a user terminates input requests, XEDIT reacts in the following manner:

- The editor stops processing the command that it was executing.
- XEDIT sends a double question mark to the user, indicating that a new XEDIT command should be issued. However, if a multiple command is being executed, XEDIT continues processing with the next command in the list.
- The file pointer is positioned at the last line that was input.

XEDIT COMMAND PARAMETERS

Throughout this manual, many XEDIT commands contain the following replaceable parameters:

n = how many file lines or string-lines should be affected by the command in question. The value entered as n must be an integer and can not exceed 99999. Its default value is 1. Users can enter an asterisk (*) for n when they want the command executed until the file pointer reaches the end-of-information (the END OF FILE message). A string-line is defined as a line which contains at least one occurrence of a specified string.

m = same as n parameter except it specifies the number of occurrences of an entity such as a string, end-of-file mark or end-of-record mark.

ln = line number that identifies which file line will be edited. The value of ln must be integer and can not exceed 99999. By default, its value is 1.

FILE POINTER CONVENTIONS

Conventions pertaining to the movement of the XEDIT file pointer are discussed in the subsequent section on "Positioning the File Pointer," which also lists the various XEDIT commands which manipulate the pointer.

POSITIONING THE FILE POINTER

During its editing, XEDIT maintains a pointer that is positioned at (that is, points to) the line in the user's file which is currently being processed. When XEDIT is called into execution, the pointer initially is positioned at the first line in the user's file.

When the user issues an XEDIT command, the pointer will be advanced to the line in the file that is affected by the execution of the command. Then, when a subsequent command is issued, the new command's execution will start at the new pointer position. In other words, each command will not automatically begin its execution from the first line in the file.

As a general rule, if the user is processing in verify mode, the pointer is usually positioned at the last line which is displayed.

Additionally, when the execution of a command causes XEDIT to read the end-of-information mark, the following actions occur:

- The following message is listed at the user's terminal:
- END OF FILE
- Further processing of the command is terminated.
 - The pointer is repositioned to the beginning of the user's file.

In summary, the following file pointer conventions apply to XEDIT.

<u>Condition</u>	<u>Pointer Position</u>
1. XEDIT is initially called.	Beginning-of-file
2. A command is executed.	Last position in file affected by the completed execution of the command (in verify mode, usually at the last line which is displayed)
3. Command execution causes end-of-information (the END OF FILE message) to be reached.	Beginning-of-file
4. Interruption of XEDIT output.	At (or one line after) the last line being <u>processed</u> , not necessarily the last line printed.

POINTER MOVEMENT BY COMMAND PREFIXING

Users can vary pointer position before the execution of an XEDIT command by prefixing the command with either a slash or a caret (or up arrow). Accordingly, the following conventions apply:

<u>Prefix Character</u>	<u>Function</u>
/	Advances the pointer one line before processing the prefixed command in all cases.
^ or ↑	Repositions the pointer to the beginning-of-file before processing the prefixed command. This character is octal 76.

POINTER MOVEMENT COMMANDS

The following XEDIT commands control file pointer movement and are discussed in detail throughout this section.

Locating Lines Via Line Numbers (Ln Command)

Users can advance the pointer to a line identified by a specified line number by simply entering the line number. After the user issues the command, XEDIT begins its search for the specified line number at the current pointer position. The search for the line number is circular (wrap-around). Thus, the top of the file may be passed in order to position the pointer at the specified line. XEDIT terminates the command's execution by positioning the pointer at a line having the specified line number. However, if no line number in the file matches the specified ln value, then XEDIT positions the pointer at the line with the next closest (and higher) line number. To combine the ln command with other commands, see General XEDIT Conventions on page 2-4.)

Format

ln **CR**

where:

ln = line number which identifies the line
that the user wants to locate.

Example

```
?? 120
00120 EXT 6533
?? 140
00150 Q.E. SMITH
?? 190
00190 P.T. BEE
?? 170
00170 POB 55 ZIP 99107
```

Example Explanation: In the first entry, the user issues an ln command to locate line 120. XEDIT responds by verifying that the pointer is positioned at line 120. Subsequently, the user simply enters a line number (140). XEDIT responds by indicating the pointer is positioned at line 150. This deviation occurs because line 140 does not exist in this file and the pointer is automatically positioned at the next closest (highest) line number. In his third entry, the user issues an ln command to locate line 190. In response, XEDIT verifies that the pointer rests at line 190. In the final entry, the user issues an ln command to locate line 170. XEDIT automatically "backs up" to line 170 and prints the line.

Locating Lines Via Specified Strings (LOCATE Commands)

Users can advance the pointer to a line which contains a specific string of alphanumeric characters by issuing a LOCATE command. LOCATE commands take three different forms: One form applies when the user wants the search to be based on a single string criteria; another form is employed when the user wants XEDIT to find a line with one string followed by a second string; a third form is used to find a line with one string not followed by a second string.

When the user wants to locate a line that contains one particular string, the following command should be issued:

Format

LOCATE/string/n (CR)
 or
 L/string/n (CR)

Example

```
?? LOCATE/ZIP/
00110 1544 WILSHIRE ST ZIP 55722
?? /LOCATE/ZIP/2
00170 POB 55 ZIP 55703
00200 8710 14TH ST ZIP 55713
?? L/8711/0
STRING NOT FOUND
```

where:

string = string of alphanumeric characters which XEDIT will attempt to locate.
 NOTE: if the user omits the terminating delimiter, XEDIT assumes one should appear after the last non-blank character.

n = user wants the pointer positioned to the nth line which meets the specified string criteria. Highest allowable value = 99999; default value = 1 (that is, user wants to locate the first line which contains the specified string). If n=0 and the string is not in the line, the pointer position remains the same, and the message "STRING NOT FOUND" is issued.

Explanation: In the first entry, the user issues a LOCATE command to position the pointer to the first line in the file containing ZIP. XEDIT verifies that line 110 is located.

In the second entry, the user issues a slash-prefixed LOCATE command to advance the pointer one line before the search. XEDIT verifies that the pointer is finally positioned at line 200. In addition, it lists line 170 which was an intermediate line that met the ZIP string criteria.

In the third entry, the user issues an abbreviated LOCATE command with zero count parameter. The search for the string 8711 is to be restricted to just the line at the current pointer position (line 00200) since the zero parameter was used. In this example the string 8711 was not found on that line.

When verify mode is in effect and the user issues a LOCATE command, XEDIT performs the following actions:

- It lists the line to which the pointer is finally positioned.
- If the user-entered n parameter is greater than 1, XEDIT lists every intermediate line which meets the specified string criteria.

When the user wants to locate a line that contains one string followed by a second string, the following command should be issued:

Format

LOCATE/string1...string2/n (CR)
 or
 L/string1...string2/n (CR)

Example

```
?? LOCATE/WILSHIRE...ZIP 55711/
END OF FILE
?? LOCATE/WILSHIRE...ZIP 55722/
00100 1544 WILSHIRE ST ZIP 55722
??
```

where:

string1...string2 = string1 followed by string2. If the user omits the terminating delimiter, XEDIT assumes one should appear after the last nonblank character.

n = user wants the pointer positioned to the nth line which meets the criteria established by the string1...string2 entry. Highest allowable value = 99999; default value = 1 (that is, user wants to locate the first line containing both string1 and string2). If n=0 and the string1...string2 string is not in the line, the pointer position remains the same and an informative message is issued.

When the user wants to locate a line that contains one string but not followed by another string, the following command should be issued:

Format

LOCATE/string1---string2/n Ⓢ

or

L/string1---string2/n Ⓢ

where:

string1 = the string of alphanumeric characters which XEDIT will attempt to locate in a single line which is not followed by string2. String1 can be null (that is, not specified), in order to locate n lines which do not contain string2.

string2 = the string of alphanumeric characters that the user does not want to find in the same line as a line with string1 before it.

Explanation: In the first entry, the user wants the pointer positioned at the first line in the file which contains WILSHIRE followed by ZIP 55711. XEDIT responds by indicating it reached the end-of-file without locating that specific combination of strings.

After reaching the end-of-file, XEDIT repositions pointer to the beginning-of-file. User issues a second LOCATE command to place pointer at the first file line which contains both WILSHIRE and ZIP 55722. XEDIT responds by verifying the pointer now is located at line 110, the first line which satisfies both string criteria.

Example

```
?? LOCATE/WILSHIRE/  
00010 3780 WILSHIRE AVE ZIP 55722  
?? ↑LOCATE/WILSHIRE---AVE/  
00110 1544 WILSHIRE ST ZIP 55722  
??
```

Explanation: The user has a file of addresses and wants to find Wilshire Street. In the first entry, the user attempts to locate Wilshire Street by specifying only the string WILSHIRE and XEDIT responds with a line containing WILSHIRE AVE. Since this is not what the user was interested in, the user enters a second LOCATE command with the up-arrow (↑) prefix to begin the search again from the top of the file. The user specifies with the WILSHIRE---AVE sequence that XEDIT should look for a line that contains WILSHIRE but not followed by AVE since the user wants Wilshire Street, not Wilshire Avenue.

n = user wants the pointer positioned to the nth line which meets the criteria established by the string1---string2 entry. Highest allowable value = 99999; default value = 1 (that is, the user wants to locate the first line that contains string1 but not followed by string2). If n=0 and string1 is not in the line, the pointer position remains the same and an informative message is issued.

Advancing and Reversing the Pointer (NEXT Command)

When users want to advance the pointer (from its current position) toward the end of the file, they enter a NEXT command in the following form:

NEXT n (CR)
or
N n (CR)

where:

n = number of lines that the pointer should be advanced. Highest allowable value = 99999; default value = 1.

When users want to reverse the pointer (toward the beginning-of-file) from its current position, they enter a NEXT command in the following form:

NEXT-n (CR)
or
N-n (CR)

where:

-n = number of lines that the pointer should be moved toward the beginning of the file.
NOTE: Once the pointer reaches the beginning-of-file position, this command's execution is terminated. Highest allowable value = 99999; default: If n is omitted or given a value of 0, no pointer movement is performed.

NOTE

Reverse pointer movements using this command are much slower than forward pointer movements.

The following example illustrates these commands:

<u>Example</u>	<u>Explanation</u>
?? PRINT 6	User issues PRINT command to list six lines.
00100 M.T. JONES	
00110 1544 WILSHIRE ST ZIP 55722	
00120 EXTENSION 6533	
00130 A.B. NEWTON	XEDIT prints six lines
00140 166 HASKELL CIRCLE ZIP 55713	
00150 EXTENSION 227	
?? NEXT-3	User issues NEXT command to reverse pointer three lines.
00120 EXTENSION 6533	
?? NEXT 2	XEDIT verifies where previous command moved the pointer.
00140 166 HASKELL CIRCLE ZIP 55713	User issues NEXT command to advance pointer two lines.
??	XEDIT verifies effect of the preceding command.

Repositioning Pointer to Top and Bottom of File (TOP and BOTTOM Commands)

Users can reposition the pointer to the beginning of the file by issuing a TOP command in the following form (see also the up-arrow prefix character on page 2-4):

TOP (CR)

or

T (CR)

In addition, users can move the pointer to the bottom of the current record in a file by issuing a BOTTOM command in the following form:

BOTTOM (CR)

or

B (CR)

The following example illustrates these commands:

<u>Example</u>	<u>Explanation</u>
?? 160	User issues line number to position pointer at line 160.
00160 Q.E. SMITH	
?? TOP	XEDIT verifies its pointer was moved to line 160.
?? PRINT	User issues TOP command to reposition pointer to the first line in the user's file; no automatic verification.
### NAMES/ADDRESSES ARE FICTITIOUS ###	
?? BOTTOM	User issues PRINT command.
00210 EXT 18	XEDIT lists the line to which the pointer is currently positioned (that is, the first line in the file).
??	User issues BOTTOM command to move pointer to the last line in the current record of the file.
	XEDIT verifies where the pointer is now positioned (that is, line 210, the last line in the file).

Locating "Bad" Lines (FBADL Command)

When users want to locate a specific number of "bad" lines, they can issue FBADL commands. In this instance, "bad" lines are defined as lines which do not begin with a line number. This command is most useful when the user is operating in verify mode since the "bad" lines are listed in this situation. The following command format is valid for entering FBADL commands:

<u>Format</u>	<u>Example</u>
FBADL <u>n</u> (CR)	?? FBADL*
or	### NAMES/ADDRESSES ARE FICTITIOUS ###
FBL <u>n</u> (CR)	END OF FILE
	??

where:

n = number of "bad" lines which the user wants to locate. An asterisk (*) should be entered when the user wants to locate every "bad" line in the file. Highest allowable value = 99999; default value = 1. NOTE: The file pointer will be positioned at the last "bad" line encountered unless END OF FILE is encountered. A value = 0 assumes n = 1.

Explanation: In first entry, user issues FBADL command to locate every "bad" line in the file (that is, the n parameter is an *). Since XEDIT is in verify mode, it lists all "bad" lines. In this file, only one "bad" line was found. XEDIT also indicates it read the end-of-information mark.

Listing File Lines (PRINT Command)

Users can list lines from their files by issuing PRINT commands. This command begins its execution at the current pointer position. When the PRINT command terminates its execution, the pointer is positioned at the last line that is printed. PRINT commands take the following form:

<u>Format</u>	<u>Example</u>
PRINT <u>n</u> (CR)	?? PRINT
or	00160 Q.E. SMITH
P <u>n</u> (CR)	?? PRINT 4
	00160 Q.E. SMITH
	00170 POB 55 ZIP 55703
	00180 EXT 8837
	00190 P.T. BEE
	?? PRINT*
	00190 P.T. BEE
	00200 8710 14TH ST ZIP 55713
	00210 EXT 18
	END OF FILE
	?? PRINT*
	### NAMES/ADDRESSES ARE FICTITIOUS ###
	00100 M.T. JONES
	00120 EXT 6533
	00160 Q.E. SMITH
	00170 POB 55 ZIP 55703
	00180 EXT 8837
	00190 P.T. BEE
	00200 8710 14TH ST ZIP 55713
	00210 EXT 18
	END OF FILE
	??

where:

n = number of lines which the user wants to print. Highest allowable value = 99999; default value = 1. When the user wants to print every line in the file from the current pointer position to END OF FILE, the n parameter should contain an *. A value = 0 assumes n = 1.

Example Explanation: In the first entry, the user issues a PRINT command to list the line to which the pointer is currently positioned. In response, XEDIT prints line 160. In the second PRINT entry, the user issues a PRINT command to list four lines, beginning at the current position (that is, line 160). Accordingly, XEDIT lists four lines. In third PRINT command, the user wants to list all lines in the file that fall between the current pointer position and the end-of-file mark. XEDIT replies by listing lines 190 to 210. In addition, XEDIT indicates that it has read END OF FILE. Finally, in the last entry, the user issues a PRINT command to list the entire file. This occurs because the pointer is automatically repositioned to the beginning-of-file once END OF FILE is encountered. XEDIT responds by printing all lines in the file.

STRING EDITING

XEDIT users can modify particular strings of alphanumeric characters that appear within a file line by issuing string editing commands. The following section describes these commands in detail.

ADDING STRINGS TO THE END OF A LINE (ADD COMMAND)

Users can append a particular string to the end of an existing file line by issuing ADD commands. Accordingly, after users issue an ADD command, XEDIT proceeds to send a single question mark (?) to indicate that the user should enter the specific string. XEDIT will then append the specified string to the end of the line designated by the current pointer position (after the last nonblank character in that line). ADD commands take the following format.

<u>Format</u>	<u>Example</u>
ADD <u>n</u> (CR)	?? 160ADD
? <u>string</u> (CR)	00160 Q.E. SMITH
or	? JR
A <u>n</u> (CR)	00160 Q.E. SMITH JR
? <u>string</u> (CR)	??

where:

n = number of consecutive lines to which the specified string should be appended. Highest allowable value = 99999; default value = 1; users can enter an * for n when they want the string to be appended to every line in the file between the current pointer position and END OF FILE. A value = 0 assumes n = 1.

string = string of alphanumeric characters which the user wants to append; the string should be entered after XEDIT responds to the ADD command with a single ?.

Explanation: User positions pointer to line 160 and issues an ADD command. XEDIT responds by verifying this position. By default, this command specifies that the next line of user input should be appended to the end of the line designated by the current pointer position. XEDIT sends a single ? to inform the user that he should now enter the string which he wants appended to line 160. User reacts by entering a blank space and then the characters JR as his appendable string. XEDIT automatically verifies how it added the specified string (JR) to line 160.

REPLACING, DELETING AND INSERTING STRINGS BY CONTEXT (CHANGE AND CHANGES COMMANDS)

CHANGE and CHANGES commands enable XEDIT users to: 1) replace the contents of one string with a different string, 2) delete strings from file lines, and 3) insert strings at the beginning of a line. If the user wants to restrict what columns the change should occur in, the windowing feature can be used with the CHANGE and CHANGES commands (see WMARGIN command). If the user prefers to change strings by visual character-by-character alignment instead of the context method, see the MODIFY command.

Replacing Strings by Context

Users can replace one string with a different string by issuing a CHANGE or CHANGES command in the following format. In this instance, the edited string and the original string can be of arbitrary length and character content.[†] In addition, the search for the first specified string starts at the line designated by the current pointer position.

<u>Format</u>	<u>Example</u>
CHANGE/ <u>string1</u> / <u>string2</u> /n (CR)	?? PRINT*
or	### NAMES/ADDRESSES ARE FICTITIOUS ###
C/ <u>string1</u> / <u>string2</u> /n (CR)	00100 M.T. JONES
or	00110 1544 WILSHIRE ST ZIP 55722
C/ <u>string1a...</u> <u>string1b</u> / <u>string2</u> /n (CR)	00120 EXT 6533
or	00160 Q.E. SMITH
CHANGES/ <u>string1</u> / <u>string2</u> /m (CR)	00170 POB 55 ZIP 55703
or	00180 EXT 8837
CS/ <u>string1</u> / <u>string2</u> /m (CR)	00190 P.T. BEE
or	00200 8710 14TH ST ZIP 55713
CS/ <u>string1a...</u> <u>string1b</u> / <u>string2</u> /m (CR)	00210 EXT 18
	END OF FILE
	?? CHANGE/EXT /853-/*
	00120 853-6533
	00180 853-8837
	00210 853-18
	END OF FILE
	?? 190
	00190 P.T. BEE
	?? CHANGE/BEE/BEAN/
	00190 P.T. BEAN
	??

where:

string1 = old string which the
user wants to replace.

string1a...string1b = similar to string1
except that string1a is
followed by string1b.
Other characters
appearing between the
two strings are included
in the string that gets
changed to string2.

[†] However, if the edited string causes the file line to exceed 160 characters, XEDIT will truncate the edited version of the line and send an informative message.

string2 = new string that replaces string1.

NOTE: string1 and string2 can be different lengths and arbitrary content.

n = number of lines containing at least one occurrence of string1 which should undergo the specified string change. XEDIT will apply the change to every appearance of string1 in a line. By default, if the user omits an n value, XEDIT will only replace every occurrence of string1 with string2 in the first line found which contains string1. Highest allowable value = 99999.

In addition, if the user wants the string change applied to every line in the file from the current pointer position to END OF FILE an * should be entered as the n parameter. If n = 0, XEDIT will not advance the pointer, and any changes will occur at the current pointer position. If n = 0 and string1 is not found, XEDIT will issue the message "STRING NOT FOUND."

NOTE: If the user fails to enter a terminating delimiter, XEDIT assumes one should appear after the last nonblank character. An informative message is also sent to the user's terminal.

m = number of occurrences of string1 or string1a...string1b. By default, if the user omits the m value, XEDIT will only replace the first occurrence of the string from the current pointer position.

Explanation: User issues a PRINT command to list entire file; XEDIT lists file. In the first CHANGE command, the user indicates that every instance of EXT should be replaced by 853 throughout the entire file. XEDIT responds by verifying which lines were affected by the preceding CHANGE command. After the user issues a command to locate line 190, he enters another CHANGE command. This second CHANGE command specifies that the name BEE should be replaced with the BEAN in the line to which the pointer is currently positioned. XEDIT verifies that this change was made.

Deleting Strings by Context

CHANGE or CHANGES commands can also be used to delete a string. To accomplish this, users simply do not enter any characters for the string2 parameter. In effect, string2 becomes a null string. The effects of these commands begin at the current pointer position and are executed on n number of file lines or m number of strings.

<u>Format</u>	<u>Example</u>
CHANGE/ <u>string1</u> // <u>n</u> (CR)	?? PRINT*
or	00100 M.T. JONES
C/ <u>string1</u> // <u>n</u> (CR)	00110 1544 WILSHIRE ST ZIP 55722
or	00120 EXT 6533
C/ <u>string1a</u> ... <u>string1b</u> // <u>n</u> (CR)	00160 Q.E. SMITH
or	00170 POB 55 ZIP 55703
CHANGES/ <u>string1</u> // <u>m</u> (CR)	00180 EXT 8837
or	00190 P.T. BEE
CS/ <u>string1</u> // <u>m</u> (CR)	00200 8710 14TH ST ZIP 55713
or	00210 EXT 18
CS/ <u>string1a</u> ... <u>string1b</u> // <u>m</u> (CR)	END OF FILE

where:

string1 = string of alpha-
numeric characters
which the user wants
to delete.

string1a...string1b = similar to string1 except
that string1a is followed
by string1b. Other
characters appearing
between the two strings
are included in the string
that gets deleted.

```
?? PRINT*
00100 M.T. JONES
00110 1544 WILSHIRE ST ZIP 55722
00120 EXT 6533
00160 Q.E. SMITH
00170 POB 55 ZIP 55703
00180 EXT 8837
00190 P.T. BEE
00200 8710 14TH ST ZIP 55713
00210 EXT 18
END OF FILE
?? CHANGE/ZIP/*
00110 1544 WILSHIRE ST 55722
00170 POB 55 55703
00200 8710 14TH ST 55713
END OF FILE
??
```

Explanation: User enters PRINT command to list the file; XEDIT responds accordingly. Then, the user issues a CHANGE command to delete every appearance of the word ZIP from every line in the file. XEDIT responds by verifying which lines were deleted by the preceding command.

n = number of lines, containing string1, that should be affected by this deletion command. Highest allowable value = 99999; default value = 1; an * indicates that the deletion should occur in every line from the current pointer position to END OF FILE. If n = 0, XEDIT will not advance the pointer, and any deletions will occur at the current pointer position. If n = 0 and string1 is not found, XEDIT will issue the message "STRING NOT FOUND" and the line is not deleted.

NOTE: If the user fails to enter a terminating delimiter, XEDIT will automatically place one immediately after the last nonblank character and issue an informative message.

m = number of occurrences of string1 or string1a...string1b. By default, if the user omits the m value, XEDIT will delete only the first occurrence of the string from the current pointer position.

Inserting Strings at the Beginning of a Line

Users can insert strings before the first (leftmost) character of a line. To accomplish this, users simply do not enter any characters for the string1 parameter. In effect, string1 becomes a null string. As in the standard CHANGE or CHANGES command formats, the effects of these commands begin at the current pointer position and are executed on n number of file lines.

<u>Format</u>	<u>Example</u>
CHANGE// <u>string2/n</u> (CR)	?? PRINT3
or	00100 M.T. JONES
C// <u>string2/n</u> (CR)	00110 1544 WILSHIRE ST ZIP 55722
or	00120 EXT 6533
CHANGES// <u>string2/n</u> (CR)	?? ↑CHANGE//EX/3
or	EX00100 M.T. JONES
CS// <u>string2/n</u> (CR)	EX00110 1544 WILSHIRE ST ZIP 55722
	EX00120 EXT 6533
	??

where:

string2 = string of alphanumeric characters which the user wants to insert before the leftmost character of a line.

n = number of lines which the user wants string2 inserted before. Highest allowable value = 99999; default value = 1; an * indicates that the insertion should occur in every line from the current pointer position to END OF FILE. If n = 0 XEDIT will not advance the pointer and any insertions will occur at the current pointer position.

Explanation: User enters PRINT command to list part of the file; XEDIT responds accordingly. Then, the user issues a CHANGE command to insert the characters EX in front of three lines. XEDIT responds by verifying which lines were affected.

MODIFYING STRINGS ON CHARACTER-BY-CHARACTER BASIS (MODIFY COMMAND)

MODIFY commands let users alter string contents on a character-by-character basis. Accordingly, the user issues a MODIFY command that references a specific line number. XEDIT responds by listing the line in question. On a subsequent line, XEDIT prints a single question mark to indicate that the user should enter his MODIFY directives, which specify how the line should be altered. These directives should be entered immediately under that portion of the printed file line which the user wants to modify.[†] Table 2-1 lists valid MODIFY directives.

NOTE

If the user's MODIFY directives cause the edited line to exceed 160 characters, XEDIT truncates the line and sends an informative message (see appendix A).

Finally, once the user performs a carriage return to transmit the MODIFY directives to XEDIT, the editor responds by verifying how the file line was actually modified. (However, if the user does not enter any directives in the directives line, XEDIT neither performs its verification nor makes changes.) The following format applies to this command.

<u>Format</u>	<u>Example</u>
MODIFY (CR)	?? 10MODIFY
or	10XTHIS STRING TO BE MODITFD
M (CR)	? & ↑IS THE # D # ↑IE#
	10 THIS IS THE STRING TO BE MODIFIED
	??

Explanation: In the first entry, the user issues a MODIFY command with the line number prefix to modify line 10. Accordingly, XEDIT lists line 10. In the third line, the user issues MODIFY directives to alter the line appearing above it. Finally, in the last line, XEDIT verifies how the user's MODIFY directives changed line 10.

[†] Since correct alignment is crucial for the successful execution of this command, users should be certain their terminal carriage return operates at a correct speed. See the note on page 2-1 for information pertaining to the NOS ROUT command.

TABLE 2-1. MODIFY DIRECTIVES

Directive	Function	Example
↑string#	Causes the string of alphanumeric characters between the ↑ and the # to be inserted in front of the character pointed to by the ↑. On Teletypewriter units and CDC 713's, a # character corresponds to a sharp (upper case 3). A ↑ character corresponds to an up arrow (upper case N) on Teletypewriter units or a carat on CDC 713's. Issuing a ↑# combination results in placing a # in the file line. NOTE: If an & or a ↑ is entered within the string, that special character will be treated as a normal character (instead of as a directive). Additionally, if a ↑ appears before a string but without the terminating #, XEDIT assumes the # should appear after the last nonblank character in the directives line.	?? 200MODIFY 00200 8710 14TH ST ? ↑ SOUTH# 00200 8710 SOUTH 14TH ST ??
↑	When a ↑ appears alone (that is, without any other trailing characters), XEDIT inserts one blank space in front of the character immediately above the ↑. On Teletypewriter units, the ↑ character corresponds to an up arrow (upper case N), while it corresponds to a caret on CDC 713's.	?? 110MODIFY 00110 1544 WILSHIREST ? ↑ 00110 1544 WILSHIRE ST ??
#	When a # appears alone (that is, without a preceding ↑), it causes the character above it to be deleted. XEDIT automatically closes up the space left by the deletion.	?? 170MODIFY 00170 POB 55 ZIP 55703 ? # 00170 PO 55 ZIP 55703 ??
blank space	Leaves the character above it unchanged.	?? 200MODIFY 00200 8710 SOUTH 14TH ST ? ??
&	Replaces the character above it with a blank space. Unlike the # directive, there is no automatic closing up of space when an & directive is issued. On CDC 713 and Teletypewriter units, an & corresponds to an ampersand (upper case 6).	?? 100MODIFY 00100 M.T.RJONES ? & 00100 M.T. JONES ??
other alphanumeric characters	Replace the characters above them with the characters which appear in the directives line. Users should not enter characters which do not exist with the normal 64-character TTY character set.	?? 160MODIFY 00160 Q.E. SMITH ? Y E 00160 Q.E. SMYTHE ??

MODIFYING STRINGS ON BASIS OF COLUMN NUMBERS (QMOD AND YQMOD COMMANDS)

Users can modify portions of their files on the basis of column numbers by issuing QMOD and YQMOD instructions. In the case of QMOD commands, the user issues the command to indicate that a specific number of lines (starting at the current pointer position) should be modified. In response, XEDIT prints a sequential list of column numbers. On a subsequent line, it transmits a single question mark to inform the user that this blank line (appearing under the column number line) constitutes the directives line. Consequently, the user should enter appropriate MODIFY directives (see table 2-1) to specify exactly how the file line(s) should be modified. In this instance, the appropriate directive should be entered directly under the numbered column that should be modified.[†]

After the user enters a carriage return to transmit the directives to XEDIT, the editor verifies how the modification(s) affects the line(s) in question. If the user does not submit any entry in the directives line, XEDIT will not verify this action and will not execute the command. The format for entering a QMOD command is:

Format

QMOD n **CR**

or

QM n **CR**

where:

n = number of lines (starting at the current pointer position) that should be modified. Highest allowable value = 99999; default value = 1. If users want to modify every line in the file between the current pointer position and END OF FILE an * should be entered for this parameter. A value = 0 assumes n = 1.

Example

```
?? PRINT*
00100 PART NO= 749322   QUAN=  757   VALUE=  945.50
00110 PART NO= 749323   QUAN= 1298   VALUE=   23.95
00120 PART NO= 749324   QUAN=  446   VALUE=  138.05
00130 PART NO= 749325   QUAN=   15   VALUE= 1650.50
00140 PART NO= 749326   QUAN=  376   VALUE=  182.75
END OF FILE
?? QMOD*
   0         1         2         3         4         5         6
123456789012345678901234567890123456789012345678901234567890
?
   00100 PART NO= 7493-22  QUAN=  757   COST=  945.50
   00110 PART NO= 7493-23  QUAN= 1298   COST=   23.95
   00120 PART NO= 7493-24  QUAN=  446   COST=  138.05
   00130 PART NO= 7493-25  QUAN=   15   COST= 1650.50
   00140 PART NO= 7493-26  QUAN=  376   COST=  182.75
END OF FILE
??
```

[†] Since correct alignment is crucial for the successful execution of QMOD and YQMOD commands, users should be certain their terminal carriage return operates at a correct speed. See the note on page 2-1 for information pertaining to the NOS ROUT command.

Example Explanation: In the first entry, the user issues a PRINT command to list the entire file. XEDIT reacts by listing the file (in this case, a relatively structured inventory file). In the next entry, the user issues a QMOD command indicating that his subsequent MODIFY directives should affect every file (that is, n = *). XEDIT responds by printing a sequence of column numbers and sending a single question mark for the user's directives input. Then, the user proceeds to enter appropriate MODIFY directives into the directives line (that is, he inserts a hyphen in front of column 20 and modifies the content of column 42 so that the word COST replaces the word VALUE). XEDIT reacts by verifying the modified content of this file.

When users want to modify a line (or lines) on the basis of column numbers, but do not want XEDIT to print the column numbers above the directives line, they should issue YQMOD commands. The conventions and rules pertaining to YQMOD are identical to those governing QMOD. The format for issuing YQMOD commands is:

Format

YQMOD n **CR**

or

YQM n **CR**

where:

n = number of lines (starting at the current pointer position) that should be affected by the subsequent MODIFY directives. Highest allowable value = 99999; default value = 1. To modify all lines from the current pointer position to END OF FILE, enter an *. A value = 0 assumes n = 1.

Example

```
?? PRINT*
00100 PART NO= 7493-22      QUAN=  757      COST=  945.50
00110 PART NO= 7493-23      QUAN= 1298      COST=   23.95
00120 PART NO= 7493-24      QUAN=  446      COST=  138.05
00130 PART NO= 7493-25      QUAN=   15      COST= 1650.50
00140 PART NO= 7493-26      QUAN=  376      COST=  182.75
END OF FILE
?? YQMOD*
?      ↑ FACTORY#
00100 FACTORY PART NO= 7493-22      QUAN=  757      COST=  945.50
00110 FACTORY PART NO= 7493-23      QUAN= 1298      COST=   23.95
00120 FACTORY PART NO= 7493-24      QUAN=  446      COST=  138.05
00130 FACTORY PART NO= 7493-25      QUAN=   15      COST= 1650.50
00140 FACTORY PART NO= 7493-26      QUAN=  376      COST=  182.75
END OF FILE
??
```

Example Explanation: In the first entry, the user issues a PRINT command to list the entire contents of a file. XEDIT proceeds to list the file (in this case, a relatively structured file of inventory information). As a second command, the user issues a YQMOD command and specifies that the subsequent MODIFY directives should apply to every line in the file (that is, n = *). After XEDIT sends a single question mark, the user enters one MODIFY directive. In this instance, he counts over seven spaces and inserts the word FACTORY in front of the information which begins in column seven (that is, in front of the word PART). Subsequently, XEDIT verifies how the preceding directive affected in edited file.

STRING SEARCH CONTROL

Users can control string searches in the following ways, in addition to the several forms of a legal string search (that is, /string/ or /string1...string2/ or /string1---string2/ or /---string2/):

- TRIM Tells XEDIT to ignore trailing blanks on string search commands such as LOCATE.
- WMARGIN Defines columns which restrict the scope of all string search commands when used with the "W" and "A" command postfix characters.

IGNORING TRAILING BLANKS (TRIM COMMAND)

Users can tell XEDIT to ignore trailing blanks on commands that involve string searches by issuing the TRIM command. The TRIM command can be issued in any one of three forms. The TRIM mode switch is either toggled or set on or off depending on which of the following forms the user enters:

<u>Format</u>	<u>Function</u>
TRIM (CR) or	Toggles between TRIM mode on and TRIM mode off. That is, if TRIM mode is off, issuing a TRIM command will turn it on and trailing blanks will be ignored on string search commands. If TRIM mode is on, issuing a TRIM command will turn it off. <u>Default</u> value is TRIM mode off (trailing blanks are not ignored).
TRIM+ (CR) or	Turns TRIM mode on -- ignores trailing blanks.
TRIM- (CR)	Turns TRIM mode off -- uses trailing blanks.

The commands that are affected by the TRIM mode on state are:

CHANGE	COPY	DELETE	OCTCHANGE
CHANGES	COPYD	LOCATE	

NOTE

Lines that are entirely blank will not be searched at all when the user is in TRIM mode.

Example

```
?? PRINT*
*EXECUTIVE MANAGEMENT*
EX100 A.B. KRAMER HQR24130
EX130 J.J. JOHNSON SWP19130
EX120 B.C. MILLER HQR44130
END OF FILE
?? LOCATE/130 /
EX100 A.B. KRAMER HQR24130
?? TRIM
?? ↑LOCATE/130 /
EX130 J.J. JOHNSON SWP19130
??
```

Example Explanation: After a listing of the file is obtained, the user attempts to find manager number 130 but instead finds manager number 100 since his facility number HQR24130 happened to also have the number 130 in it. To correctly locate manager number 130, the user issues a TRIM command to tell XEDIT to ignore the trailing spaces (blanks) to the right of all facility numbers such as HQR24130. The LOCATE command now finds the right line with 130 in it that was desired, namely EX130.

DEFINING A WINDOW (WMARGIN COMMAND)

Users can restrict the scope of all string searches to a specified range of columns by issuing the WMARGIN command. Once the window columns have been defined, the user can ask for the window whenever desired. The user makes the window request by simply appending either a "W" or "A" postfix character to the end of any of the string search commands (for example, CHANGEW and LOCATEW). To set the left and right window margin columns use the WMARGIN command as follows:

Format

WMARGIN lm rm (CR)
or
WM lm rm (CR)

where:

lm = column position setting of the left
window margin. Initial value = 1.
rm = column position setting of the right
window margin. Initial value = 160.

$1 \leq \underline{lm} \leq \underline{rm} \leq 160$

Example

```
?? P*
DEPT4208 EMP927 KRAMER, SCOTT W.
DEPT4208 EMP208 JOHNSON, JACK J.
DEPT4208 EMP742 MILLER, RICK E.
DEPT4208 EMP698 SCOTT, FRANK R.
END OF FILE
?? WMARGIN 10 15
?? LW/208/
DEPT4208 EMP208 JOHNSON, JACK J.
?? WM 17 17
?? ↑LA/SCOTT/
DEPT4208 EMP698 SCOTT, FRANK R.
??
```


Notice that when using the anchor "A" postfix character, the only requirement is that the first character of the string be found within the window.

Once a window has been defined, the user can request that the window restraints be used by appending either a "W" or "A" postfix character to the end of any of the string search commands. Thus legal commands would be:

The corresponding command abbreviations are also allowed to have a postfix character.

The "A" or anchor postfix character requires that only the first character of the string specified by string or string1...string2 or string1---string2 must reside within the window margins, otherwise, the string is not found by XEDIT. All other characters can extend beyond the windowed area.

ABCDEFGH I J K L M

← window →

LW/CDEF...IJ/ fails

Locate within the window string 'EF' not followed by 'H':

LW/EF---H/ fails

Locate within the window the anchor of string 'EF' followed by 'KLM':

LA/EF...KLM/ succeeds

Locate within the window the anchor of string 'KLM':

LA/KLM/ succeeds

Locate within the window the anchor of string 'EF' not followed by 'M':

LA/EF---M/ fails

Locate within the window the anchor of string 'CDEF' followed by 'IJ':

LA/CDEF...IJ/ fails

Locate within the window the anchor of string 'DEF' not followed by 'ZZ':

LA/DEF---ZZ/ succeeds

Test the window to see if it does not include string 'LM':

LW/---LM/ succeeds

or

LA/---LM/ succeeds

Change within the window string 'EF' followed by 'IJ' to 'XX':

CW/EF...IJ/XX/ produces 'ABCDXXKLM'

Change within the window string 'JKL' to 'XX'; do not move pointer:

CW/JKL/XX/0 produces 'STRING NOT FOUND'

Change the anchored string 'JKL' to 'XX':

CA/JKL/XX/ produces 'ABCDEFGHIXXM'

Insert string 'XX' at the beginning of the window:

CW//XX/ produces 'ABCXXDEFGHIJKLM'

or

CA//XX/ produces 'ABCXXDEFGHIJKLM'

CAUTION

All of the above examples assume that the window 4 through 11 (DEFGHIJK) is defined for a one-line file with text ABCDEFGHIJKLM. If the user wants to run these examples sequentially, the RESTORE command should be issued after each example is executed.

LINE EDITING

Users can modify entire lines under XEDIT by issuing a series of line editing commands. These commands are discussed in the following section.

DELETING LINES (DELETE COMMAND)

Users can delete entire lines from their files by employing DELETE commands in four different ways:

- By deleting a sequence of lines starting from the current pointer position.
- By deleting selective lines on the basis of specified string criteria.
- By deleting a single line with a specified string without moving the line pointer if the string is not found.
- If the user wants to restrict what columns the specified string should occur in before the line deletion takes place, the windowing feature can be used with the DELETE command (see WMARGIN command).

See also the COPYD command for additional ways of deleting lines.

Once a DELETE command has been executed, the file pointer is positioned at the line that appears after the last deleted line. The forms of the DELETE command are:

Format

DELETE n (CR)
or
D n (CR)

where:

n = number of lines (starting at the current pointer position) which should be deleted. Highest allowable value = 99999; default value = 1. When users want to delete all lines which appear between the current pointer position and END OF FILE, they should enter an *. A value = 0 assumes n = 1.

Example

```
?? 130
00130 B.P.PEEPERS
?? DELETE 3
00130 B.P.PEEPERS
00140 116 WEST ELM DRIVE ZIP 55648
00150 EXT 3222
?? 190
00190 P.T. BEE
?? XDELETE*
END OF FILE
??
```

Explanation: User positions pointer to line 130, then issues a DELETE command which specifies that (starting at the current pointer position) three lines should be deleted. XEDIT responds by verifying which three lines are deleted.

In the next sequence, the user positions pointer to line 190. Then, he issues a DELETE command to remove all lines that appear between line 190 and END OF FILE. XEDIT indicates the END OF FILE was encountered, but does not verify which lines were deleted since the X prefix was used.

Format

DELETE/string/n (CR)

or

D/string/n (CR)

where:

string = string of alphanumeric characters that XEDIT will use as a criteria for locating which lines to delete.

n = number of lines (starting at the current pointer position) that should be deleted if they meet the user-specified string criteria. Highest allowable value = 99999; default value = 1. Users can enter an * if they want to delete every line that appears between the current pointer position and END OF FILE as long as it meets the specified string criteria. If n = 0 and the string is not in the line, the pointer position remains the same, an informative message is issued, and the line is not deleted.

Format

DELETE/string1...string2/n (CR)

or

D/string1...string2/n (CR)

where:

string1...string2 = two strings of alphanumeric characters (existing within a single line) that establish the string criteria upon which XEDIT will base its deletion. These specified strings will constitute the deletion criteria even though they may be separated in a line by an indeterminate number of other characters or phrases.

Example

```
?? DELETE/EXT/*
00120 EXT 6533
00150 EXT 227
00159 EXT 5339
00180 EXT 8866
00210 EXT 18
END OF FILE
??
```

Explanation: User issues a DELETE command to delete every line in the file which contains the word (that is, string) EXT.

NOTE

This assumes that the initial pointer position was the beginning-of-file. XEDIT responds by verifying which particular lines were deleted.

Example

```
?? PRINT*
00100 M.T. JONES
00110 1544 WILSHIRE ST ZIP CODE 55722
00120 EXT 6533
00130 A.B. MACDONALD
00140 1313 LEMONTREE AVE ZIP CODE 55722
00150 EXT 5339
00160 T.G. SLATER
00170 322 WILSHIRE ST ZIP CODE 55723
00180 EXT 67
00190 R.C. CARTER
00200 6100 WILSHIRE ST ZIP CODE 55722
00210 EXT 1101
END OF FILE
?? DELETE/WILSHIRE ST...55722/*
00110 1544 WILSHIRE ST ZIP CODE 55722
00200 6100 WILSHIRE ST ZIP CODE 55722
END OF FILE
??
```

n = number of lines that the user wants to delete if they meet the specified string1...string2 criteria. Highest allowable value = 99999; default value = 1. If the user wants to delete every line which falls between the current pointer position and END OF FILE and meets the string1...string2 criteria, an * should be entered for this parameter. If n = 0 and the string1...string2 string is not in the line, the pointer position remains the same, an informative message is issued, and the line is not deleted.

Format

DELETE/string1---string2/n (CR)

or

D/string1---string2/n (CR)

where:

string1---string2 = delete n lines which contain string1 but not followed by string2. String1 can be null (that is, not specified), in order to delete n lines which do not contain string2.

n = user wants the pointer positioned to the nth line which meets the criteria established by the string1---string2 entry. Highest allowable value = 99999; default value = / (that is, the user wants to delete n lines which contains string1 but not followed by string2). If n = 0 and string1 is not in the line, the line is not deleted, the pointer position remains the same, an informative message is issued, and the line is not deleted.

Explanation: User issues a PRINT command to list the entire file; XEDIT complies. Then, he issues a DELETE command to remove every line in the file that contains WILSHIRE ST and 55722. XEDIT responds by verifying which lines were deleted.

Example

```
?? DELETE/WILSHIRE/
00010 3780 WILSHIRE AVE ZIP 55722
?? RESTORE
?? DELETE/WILSHIRE---AVE/
00110 1544 WILSHIRE ST ZIP 55722
??
```

Explanation: The user has a file of addresses and wants to delete Wilshire Street. In the first entry, the user attempts to delete Wilshire Street by specifying only the string WILSHIRE and XEDIT responds by deleting a line which contains WILSHIRE AVE. Since this is not what was desired, the user issues a RESTORE command to reinstate the accidentally deleted line. The user issues a second DELETE command, this time specifying that a line with the string WILSHIRE but not followed by AVE should be deleted.

REPLACING LINES (REPLACE COMMAND)

REPLACE commands allow users to replace a specific number of existing file lines (starting at the current pointer position) with a same number of substitute lines. After the user issues the command, XEDIT transmits a single question mark to inform the user that he should enter his substitute line. This procedure is repeated until the user has submitted the number of lines specified in the original REPLACE command.[†] These commands take the following form:

<u>Format</u>	<u>Example</u>
REPLACE <u>n</u> (CR)	?? 170
or	00170 POB 55 ZIP 55703
R <u>n</u> (CR)	?? REPLACE 2
	? 00170 18 PARK PLACE APT 111 ZIP 55704
	? 00180 EXT 8866
	??

where:

n = number of lines starting at the current pointer position which the user wants to replace. Highest allowable value = 99999; default value = 1. When users want to replace every line from the current pointer position to END OF FILE, they should enter an * for this parameter value. A value = 0 assumes n = 1.

Explanation: User positions pointer at line 170, then issues a REPLACE command indicating that two lines (starting at line 170) should be replaced. XEDIT sends a single ? indicating user should enter his first replacement. User reacts by adding a modified address. Subsequently, XEDIT sends the second single ? and the user enters a modified extension number.

INSERTING NEW LINES

Users can insert entire new lines into a file, without affecting any existing lines, in three different ways:

- Issuing INSERT commands -- to insert a specific number of lines after the current pointer position
- Issuing INSERTB commands -- to insert a specific number of lines in front of the current pointer position
- Entering INPUT mode by performing a carriage return or issuing an INPUT command -- to insert any number of subsequent user-specified lines after the current pointer position

[†] If the user performs a carriage return without any accompanying entries, XEDIT exits REPLACE mode and requests that the next command be entered (a double question mark: ??). To replace with a blank line, the user enters a space and presses carriage return.

INSERT Command

INSERT commands let a user insert a specific number of entire new lines into a file immediately after the line designated by the current pointer position. Once the user submits the INSERT command, XEDIT transmits a single question mark (to indicate the user can now enter his initial line for insertion). The process of issuing the inserted lines one line at a time is repeated until the user has submitted the specified number of lines.[†] After the user transmits the final insert, XEDIT positions the pointer at the last inserted line. INSERT commands take the following form:

<u>Format</u>	<u>Example</u>
INSERT <u>n</u> <u>CR</u>	?? 120
or	00120 EXT 6533
I <u>n</u> <u>CR</u>	?? INSERT 3
	? 00130 A.B. NEWTON
	? 00140 166 HASKELL CIRCLE ZIP 55713
	? 00150 EXT 227
	?? ↑PRINT*
	### NAMES/ADDRESSES ARE FICTITIOUS ###
	00100 M.T. JONES
	00110 1544 WILSHIRE ST ZIP 55722
	00120 EXT 6533
	00130 A.B. NEWTON
	00140 166 HASKELL CIRCLE ZIP 55713
	00150 EXT 227
	00160 Q.E. SMITH
	00170 18 PARK PLACE APT 111 ZIP 55704
	00180 EXT 8866
	00190 P.T. BEE
	00200 8710 14TH ST ZIP 55713
	00210 EXT 18
	END OF FILE
	??

Example Explanation: User positions pointer to line 120. Subsequently, he issues an INSERT command to specify that three new lines should be inserted immediately after line 120 (that is, the current pointer position). XEDIT responds by transmitting a single question mark and the user reacts by entering a new line that contains an employee name. This pattern is repeated twice more as the user enters new lines containing an address and a phone extension.

To check this sequence, the user issues a PRINT command with an up-arrow (↑) prefix to reposition the pointer at the top of the file. Accordingly, XEDIT prints every line in the file and the user can see where the new lines (130-150) were inserted.

[†] If the user performs a carriage return without any accompanying entries, XEDIT exits INSERT mode and requests that the next command be entered (a double question mark: ??). To insert a blank line, the user enters a space and presses carriage return.

INSERTB Command

When users want to insert a specific number of lines into a file in front of the line designated by the current pointer position, they do so by issuing INSERTB commands. Accordingly, the user enters an INSERTB command indicating how many lines should be inserted. XEDIT responds by transmitting a single question mark (to indicate the user should enter the top-most line that is part of the insert sequence). This procedure of XEDIT prompting/user entry continues until the user has issued the same number of inserted lines as he specified in the initial INSERTB command.[†] After XEDIT finishes executing an INSERTB command, the pointer remains at the original position it occupied before the command was first issued. INSERTB commands take the following form:

<u>Format</u>	<u>Example</u>
INSERTB <u>n</u> (CR)	?? 160
or	00160 Q.E. SMITH
IB <u>n</u> (CR)	?? INSERTB 3
	? 00157 A.P. MACDONALD
	? 00158 1313 LEMONTREE AVE ZIP 55722
	? 00159 EXT 5339
	?? 140PRINT6
	00140 166 HASKELL CIRCLE ZIP 55713
	00150 EXT 227
	00157 A.P. MACDONALD
	00158 1313 LEMONTREE AVE ZIP 55722
	00159 EXT 5339
	00160 Q.E. SMITH
	??

where:

n = number of lines which the user wants to insert. Highest allowable value = 99999; default value = 1. A value = 0 assumes n = 1.

Example Explanation: User positions pointer to line 160. Subsequently, he issues an INSERTB command to insert three new lines that should be placed in front of line 160. After XEDIT transmits the first single question mark, the user enters the line (that is, line 157) which he wants to appear as the top-most line of the new three-line sequence. This XEDIT-prompting/user-entry pattern is repeated twice more as the user enters lines 158 and 159. Then, the user enters a PRINT command of six lines beginning at line 140 to check how XEDIT processed the preceding insertion. As the file listing indicates, lines 157, 158, and 159 were sequentially inserted between lines 150 and 160.

INPUT Mode Entries (Carriage Return or INPUT/EDIT Commands)

When users want to insert an unspecified number of new lines into a file, they can enter INPUT mode in either of the following two ways:

1. press carriage return (only for interactive usage); or
2. enter the INPUT command (for either interactive or batch usage).

Either of these methods enable the user to enter lines for insertion into the file immediately after the current pointer position.

[†] If the user performs a carriage return without entering any accompanying data, XEDIT exits INSERTB mode and requests that the next command be entered (a double question mark: ??). To insert a blank line, the user enters a space and presses carriage return.

Carriage Return Method

The sequence of steps for the carriage return method include: 1) users enter a carriage return instead of a command, 2) XEDIT responds by printing the word INPUT, followed on a separate line by a single question mark, 3) users terminate this procedure by pressing the carriage return instead of entering a new line of information, 4) XEDIT indicates this insertion procedure is now finished by printing the word EDIT, 5) XEDIT then transmits a double question mark to indicate that users are now expected to issue an appropriate XEDIT command.

An example of entering INPUT mode using the carriage return method is shown below.

Example	Explanation
?? 120	User positions pointer to line 120.
00120 EXT 6533	
??	User presses carriage return.
INPUT	XEDIT indicates that the user can now enter appropriate new lines for insertion.
? 00121 M.C.GREENWAY	
? 00122 867 MISSION ST ZIP 55744	
? 00123 EXT 1500	XEDIT transmits a series of single question marks. User reacts to each one by entering a line for insertion.
? 00124 SPECIAL CLASS**SUPERVISOR	
? 00125 EMP: 12	
? 00126 LIST: C	
?	User presses carriage return to terminate the insertion procedure.
EDIT	XEDIT indicates that normal editing can now continue.
?? T	User issues abbreviated TOP and PRINT commands to check the insertion process.
?? P *	
### NAMES/ADDRESSES ARE FICTITIOUS ###	
00100 M.T. JONES	
00110 1544 WILSHIRE ST ZIP 55722	
00120 EXT 6533	
00121 M.C.GREENWAY	
00122 867 MISSION ST ZIP 55744	
00123 EXT 1500	
00124 SPECIAL CLASS**SUPERVISOR	
00125 EMP: 12	
00126 LIST: C	
00130 A.B. NEWTON	
00140 166 HASKELL CIRCLE ZIP 55713	
00150 EXT 227	
00170 A.P. MACDO	User terminates listing by pressing the break key.
??	

INPUT/EDIT Command Method

The user may also enter and exit INPUT mode by issuing INPUT and EDIT commands respectively instead of pressing carriage return. This is especially useful when doing batch processing with XEDIT (see appendix C on "XEDIT Batch Command Processing"). In addition, it has an advantage over the carriage return method. The user can issue a useful subset of XEDIT commands while under INPUT mode to make "spot" changes to the line just entered. Thus, if an error was made in entering a line while under INPUT mode, the user does not have to exit INPUT mode to make the correction.

To use the INPUT/EDIT method, the user must issue an escape character on the INPUT command. The escape character is subsequently used during INPUT mode as a prefix character for any XEDIT command lines which do not move the pointer. The escape character will remain in effect until the user issues another INPUT command, even if the user exits and reenters INPUT mode via a carriage return.

During input, if the user enters a line with the escape character as the first character, XEDIT will strip off the escape character and will execute the rest of the line as a command line. As long as the user remains in INPUT mode, any command which repositions the pointer is illegal (specifically DELETE cannot be used - use REPLACE instead). XEDIT requires all specified command line repetition counts (n parameter) to be zero or null, otherwise ARGUMENT ERROR is issued.

Upon completion of the escaped command sequence or if a syntax error occurred, XEDIT will automatically return the user to INPUT mode. He can then continue to enter lines of input or can exit INPUT mode.

Format

INPUT e (CR)

and

e EDIT (CR)

where:

e = escape character; any character except space or an existing command delimiter (DELIMIT). To get a comma, use (INPUT, ,). If e is the same as the tab character (DEFTAB), the tab character cannot occur in column one since it will be interpreted as an escape character.

Example

```
?? 120
00120 EXT 6533
?? INPUT $
INPUT
? 00121 M.C.GREENWAY
? $CHANGE/EEE/EE/
00121 M.C.GREENWAY
? 00122 867 MISSION ST ZIP 55744
? $EDIT
EDIT
??
```

Example Explanation: User positions pointer to line 120. The user then issues the INPUT command with the dollar sign (\$) as the escape character. XEDIT responds by indicating that it is now ready to accept lines of INPUT to be inserted after the current pointer position. In entering a line, the user notices that he spelled GREENWAY wrong so he issues a CHANGE command prefixed by the escape character (\$) to indicate that he wants the command executed, rather than being used as another line of text. XEDIT verifies that the change was made and requests more lines of input. The user enters another line of text and then exits INPUT mode to go back into normal command EDIT mode by entering the EDIT command prefixed with the escape character (\$). The escape character enables XEDIT to distinguish between text input and command line input.

Batch users should note that the use of the INPUT and EDIT commands is the only way to enter and exit INPUT mode, and the escape character must be specified with both INPUT and EDIT.

NOTE

Users of multiple commands (that is, Z or Y commands) can make use of INPUT mode by using the INPUT command as one of the commands in the list.

The following example, which uses a dollar sign as the user's command delimiter, illustrates this:

Z\$XLOCATE/R.M.SMITH/\$INPUT

INSERTING A BLANK LINE AT TOP OF A FILE (TOPNULL COMMAND)

Under certain circumstances, a user may want to insert a blank line as the first line in a file. This most commonly occurs when the user wants to delete leading record marks, but is prohibited from this since XEDIT always searches the file for the first line. This command will allow the user to write a blank line before the leading record marks. The user can then issue a DEOR command to delete the record marks. The following command format and example illustrate how to employ TOPNULL.

<u>Format</u>	<u>Example</u>
TOPNULL (CR)	OLD,BPGM
or	READY.
TN (CR)	XEDIT
	XEDIT 3.0.0
	--EOR--
	--EOR--
	?? PRINT*
	00200 PRINT "SQR EXAMPLE"
	00210 INPUT N
	00220 PRINT SQR(N)
	00230 END
	END OF FILE
	--EOR--
	--EOR--
	?? TOPNULL
	?? XDEOR2
	?? ↑DELETE
	?? PRINT*
	00200 PRINT "SQR EXAMPLE"
	00210 INPUT N
	00220 PRINT SQR(N)
	00230 END
	END OF FILE
	??

Example Explanation: User calls XEDIT to edit a file called BPGM. As his first XEDIT command, the user issues a PRINT command to list the contents of BPGM. XEDIT responds by listing the entire BPGM file which shows that it contains leading record marks. Next, the user issues a TOPNULL command to insert a blank line as the first line in the BPGM file. (The file pointer is now set to this null line.) Then, he proceeds to issue a DEOR (with an X prefix) command to delete the leading record marks. After XEDIT inserts HDR file into BPGM, the user issues a DELETE command (with a ↑ prefix) to delete the first line in the file (that is, the blank line). Finally, the user issues a PRINT command to check how the BPGM file was modified (that is, to see whether the two record marks were in fact deleted from the beginning of the BPGM file).

EDITING LINE NUMBERS

XEDIT users can issue a series of commands to modify the line numbers that appear in their files. A set of general conventions apply to all of the line numbering commands. These include:

1. Before any line numbering command is executed, the file pointer is automatically set to the beginning-of-file position.
2. After all line numbering commands are executed, XEDIT sends the following message to indicate that the specified line number editing was executed throughout the entire file:

END OF FILE

Subsequently, the pointer is automatically repositioned to the beginning-of-file.

3. If a line numbering command causes the edited file line to exceed 160 characters, then XEDIT truncates the edited line and sends an appropriate informative message.
4. No line number can ever exceed a value of 99999. If a line numbering command results in a modification which breaks this rule, the file will be restored to its original condition, an informative message will be issued, and the pointer will be positioned to the beginning-of-file.
5. Line numbering commands are primarily intended for editing line-numbered card-image text files and programs written in time-sharing FORTRAN Extended. However, these commands are unacceptable for use on BASIC programs, since BASIC branch line numbers are not modified by these commands.
6. XEDIT does not verify the effects of any line numbering command (even if verify mode is operational).

ADDING LINE NUMBERS (ADDLN AND ADDLNS COMMANDS)

Two different XEDIT commands allow users to add line numbers to a file where none previously existed:

- ADDLN commands -- apply only when a line number is added to each line[†]
- ADDLNS commands -- apply when both a line number and a trailing blank space are added to each line[†]

[†] If line numbers already exist in the user's file, ADDLN and ADDLNS will add another set of line numbers to that file.

ADDLN Command

When users want to add a line number (without a trailing space) to every line in a file, they can enter an ADDLN command in the following form:

Format

ADDLN ln n (CR)

or

ALN ln n (CR)

Example

```
?? P*
M.T. JONES
1544 WILSHIRE ST ZIP 55722
EXT 6533
A.B. NEWTON
166 HASKELL CIRCLE ZIP 55713
EXT 227
A.P. MACDONALD
1313 LEMONTREE AVE ZIP 55722
EXT 5339
END OF FILE
?? ADDLN 00010 10
END OF FILE
?? P*
00010 M.T. JONES
00020 1544 WILSHIRE ST ZIP 55722
00030 EXT 6533
00040 A.B. NEWTON
00050 166 HASKELL CIRCLE ZIP 55713
00060 EXT 227
00070 A.P. MACDONALD
00080 1313 LEMONTREE AVE ZIP 55722
00090 EXT 5339
END OF FILE
??
```

where:

ln = line number that should be assigned to the first line in the user's file. No line number can ever exceed 99999. Default = 1.

n = number by which each line number in the file will be incremented. No line number can exceed a value of 99999. Default = 1.

Example Explanation: After listing his file (notice how the first character in each file line is a blank), the user issues an ADDLN command. The first line in the file is to be given a line number of 00010 and each succeeding number is to be incremented by 10. Finally, the user issues a PRINT command to check effect of the ADDLN command.

ADDLNS Command

When users want to add both a line number and a trailing blank space to every line in a file, they should enter an ADDLNS command in the following form:

Format

ADDLNS ln n (CR)

or

ALNS ln n (CR)

Example

```
?? P*
Q.E. SMITH
18 PARK PLACE APT 111 ZIP 55704
EXT 8866
P.T. BEE
8710 14TH ST ZIP 55713
EXT 18
END OF FILE
?? ADDLNS 00200 10
END OF FILE
?? P*
00200 Q.E. SMITH
```

where:

ln = line number that should be assigned to the first line appearing in the user's file. No line number can ever exceed 99999. Default = 1.

Format

n = number by which each line number will be incremented. No line number can ever exceed 99999. Default = 1.

Example

```
00210 18 PARK PLACE APT 111 ZIP 55704
00220 EXT 8866
00230 P.T. BEE
00240 8710 14th ST ZIP 55713
00250 EXT 18
END OF FILE
??
```

Example Explanation: After listing his file (notice how all lines begin flush left, without a leading blank), the user issues an ADDLNS command. Accordingly, the first line will be assigned a line number of 00200 and each succeeding line will be given a number that is incremented by 10. Finally, user issues a PRINT command to check the results of the ADDLNS command.

DELETING LINE NUMBERS (DELETELN COMMANDS)

DELETELN commands let users remove every existing line number from a file. This command takes the following form:[†]

Format

DELETELN (CR)
or
DLN (CR)

Example

```
?? P*
00100 M.T. JONES
00110 1544 WILSHIRE ST ZIP 55722
00120 EXT 6533
00130 A.B. NEWTON
00140 166 HASKELL CIRCLE ZIP 55713
00150 EXT 227
?? DELETELN
END OF FILE
?? P*
M.T. JONES
1544 WILSHIRE ST ZIP 55722
EXT 6533
A.B. NEWTON
166 HASKELL CIRCLE ZIP 55713
EXT 227
END OF FILE
??
```

Example Explanation: After listing his file, the user issues a DELETELN command to delete every line number from that file. Then, the user issues an abbreviated PRINT command to check the results from the preceding command.

[†] DELETELN commands have no effect on a file line unless the line is preceded by a line number.

REPLACING EXISTING LINE NUMBERS (REPLACELN COMMAND)

When users want to replace an existing set of line numbers with a different set of line numbers, they can do so by entering REPLACELN commands in the following form:[†]

Format

REPLACELN ln n (CR)

or

RLN ln n (CR)

where:

ln = line number that should be assigned to the first line appearing in the user's file. No line number can exceed 99999. Default = 1.

n = number by which each line number will be incremented. No line number can exceed 99999. Default = 1.

Example

```
?? P*
00100 M.T. JONES
00110 1544 WILSHIRE ST ZIP 55722
00120 EXT 6533
00130 A.B. NEWTON
00140 166 HASKELL CIRCLE ZIP 55713
00150 EXT 227
END OF FILE
?? REPLACELN 00500 5
END OF FILE
?? P*
00500 M.T. JONES
00505 1544 WILSHIRE ST ZIP 55722
00510 EXT 6533
00515 A.B. NEWTON
00520 166 HASKELL CIRCLE ZIP 55713
00525 EXT 227
END OF FILE
??
```

Example Explanation: After listing contents of the file, the user issues a REPLACELN command. This command indicates that the first line number in the new set of line numbers should be 00500 and be incremented by 5. Subsequently, user lists the edited file to check the effects of the REPLACELN command.

PERFORMING MISCELLANEOUS EDITING

The following XEDIT commands let a user perform various miscellaneous editing functions:

- DBADL command -- to delete "bad" lines from the file
- DEOF command -- to delete end-of-file marks from the file
- DEOR command -- to delete end-of-record marks from the file
- DLBLANKS command -- to delete leading blanks from file lines
- OCTCHANGE command -- to convert display code strings
- WEOF command -- to write end-of-file marks onto a file
- WEOR command -- to write end-of-record marks onto a file

[†] REPLACELN commands will not affect any line unless it is preceded by a line number.

DELETING "BAD" LINES (DBADL COMMAND)

User can delete lines which do not begin with a line number (that is, "bad" lines) by issuing DBADL commands. Accordingly, starting at the current pointer position, XEDIT will locate and delete as many "bad" lines as are specified in the user's n parameter. When XEDIT operates in VERIFY mode, the editor automatically lists each line that is deleted.

<u>Format</u>	<u>Example</u>
DBADL <u>n</u> (CR) or DBL <u>n</u> (CR)	?? DBADL* ### NAMES/ADDRESSES ARE FICTITIOUS ### END OF FILE ??

where:

n = number of "bad" lines which the user wants to delete (starting at the current pointer position). Highest allowable value = 99999; default value = 1.

Users enter an * when they want to delete every "bad" line between the current point position and END OF FILE. A value = 0 assumes n = 1.

Explanation: User issues DBADL command to delete every "bad" line in the entire file (that is, pointer is positioned to beginning-of-file). XEDIT verifies that it deleted one line in response to the above command (that is, the file only contained one "bad" line).

DELETING AND INSERTING RECORD AND FILE MARKS (DEOR, WEOR, DEOF, WEOF COMMANDS)

XEDIT users can both insert and delete record and file marks. DEOR and DEOF commands enable users to delete these marks, while WEOR and WEOF commands allow users to insert them.

DEOR Command

By issuing DEOR commands, users can delete a specified number of end-of-record marks from the file.[†]

<u>Format</u>	<u>Example</u>
DEOR <u>m</u> (CR) or DR <u>m</u> (CR)	?? P* 00190 P.T. BEE 00200 8710 14th ST ZIP 55713 --EOR-- 00210 EXT 18 ###EXECUTIVE LISTING### END OF FILE ?? DEOR* --EOR-- END OF FILE ??

[†] If the user wants to delete an end-of-record mark that appears in front of the first line in the file, he must first issue a TOPNULL command.

where:

m = number of end-of-record marks which the user wants to delete (starting at the current pointer position). Highest allowable value = 99999; default value = 1. User should enter an * when they want to delete every end-of-record mark between the current pointer position and END OF FILE. A value = 0 assumes m = 1.

Explanation: After listing a portion of his file, the user discovers where an inadvertent end-of-record appears. Subsequently, he issues a DEOR command to delete that mark.

DEOF Command

By issuing DEOF commands, users can delete a specified number of end-of-file marks from the file.

Format

DEOF m (CR)

or

DF m (CR)

Example

```
?? P*
00190 P.T. BEE
00200 8710 14TH ST ZIP 55713
--EOF--
00210 EXT 18
###EXECUTIVE LISTING###
END OF FILE
?? DEOF*
--EOF--
END OF FILE
??
```

where:

m = number of end-of-file marks which the user wants to delete (starting at the current pointer position). Highest allowable value = 99999; default value = 1. User should enter an * when they want to delete every end-of-file mark between the current pointer position and the end-of-information. A value = 0 assumes m = 1.

Explanation: After listing a portion of his file, the user discovers where an inadvertent end-of-file appears. Subsequently, he issues a DEOF command to delete that mark.

CAUTION

The DEOF command will delete only end-of-file marks, not end-of-record marks created by the WEOF command.

WEOR Command

When users want to insert an end-of-record mark into the file in front of the current pointer position, they can do so by issuing a WEOR command in the following form:

Format

WEOR (CR)

or

WR (CR)

Example

```
?? LOCATE/EXECUTIVE/
###EXECUTIVE LISTING###
?? WEOR
?? ↑LOCATE/BEE/
00190 P.T. BEE
?? P*
00190 P.T. BEE
00200 EXT 18
--EOR--
###EXECUTIVE LISTING###
END OF FILE
??
```

Example Explanation: After positioning the pointer to the line which contains the word EXECUTIVE, the user issues a WEOF command to insert an end-of-record mark in front of that line. Then, the user issues an abbreviated PRINT command to check whether the end-of-record was correctly inserted.

WEOF Command

When users want to insert an end-of-file mark into the file in front of the current pointer position, they can do so by issuing a WEOF command in the following form:

<u>Format</u>	<u>Example</u>
WEOF (CR)	?? LOCATE/EXECUTIVE/
or	###EXECUTIVE LISTING###
WF (CR)	?? WEOF
	?? ↑LOCATE/BEE/
	00190 P.T. BEE
	?? P*
	00190 P.T. BEE
	00200 8710 14TH ST ZIP 55713
	00210 EXT 18
	--EOF--
	###EXECUTIVE LISTING###
	END OF FILE
	??

Example Explanation: After positioning the pointer to the line which contains the word EXECUTIVE, the user issues a WEOF command to insert an end-of-file mark in front of that line. Then, the user issues an abbreviated PRINT command to check whether the end-of-file was correctly inserted.

CAUTION

When using WEOF, the system will often force an end-of-record mark before the end-of-file mark.

DELETING LEADING BLANKS (DLBLANKS COMMAND)

DLBLANKS commands enable users to delete leading blanks from a specified number of lines in the file that appear between the current pointer position and END OF FILE. Blank lines are entirely deleted.

<u>Format</u>	<u>Example</u>
DLBLANKS <u>n</u> (CR)	?? P*
or	### NAMES/ADDRESSES ARE FICTITIOUS ###
DLB <u>n</u> (CR)	M.T. JONES
	1544 WILSHIRE ST ZIP 55722
	EXT 6533
	A.B. NEWTON
	166 HASKELL CIRCLE ZIP 55713
	EXT 227
	END OF FILE
	?? DLBLANKS*
	END OF FILE
	?? P*
	### NAMES/ADDRESSES ARE FICTITIOUS ###
	M.T. JONES
	1544 WILSHIRE ST ZIP 55722
	EXT 6533
	A.B. NEWTON
	166 HASKELL CIRCLE ZIP 55713
	EXT 227
	END OF FILE
	??

where:

n = the number of lines with leading blanks which the user wants leading blanks deleted (starting at the current pointer position). Highest allowable value = 99999; default value = 1. Users should enter an * when they want to delete leading blanks from all lines which contain leading blanks between the current pointer position and END OF FILE. A value = 0 assumes n = 1.

Example Explanation: After listing the file, the user issues a DLBLANKS command to delete the single blank space which appears in front of almost every line in the file. Subsequently, the user lists the file again to see how the DLBLANKS command affected the file.

CONVERTING OCTAL STRINGS (OCTCHANGE COMMAND)

OCTCHANGE commands let the user convert one set of octal code (internal display) to another set of octal code. Typically, this command is employed when a user wants to enter certain terminal line controls (for example, line-feeds and carriage returns) into his file that otherwise cannot be specified.

Format

OCTCHANGE oct1 oct2 n (CR)
or
OC oct1 oct2 n (CR)

where:

oct1 = octal (display) code that represents the existing string which will be changed within the user's file. Each display code character must be represented by an even number of octal digits. An odd number of digits (or a nonoctal character within an octal parameter) is illegal. See table 2-2 for a list of valid display codes and their various graphic counterparts.

oct2 = octal (display) code that represents the string that should replace oct1. Each display code character must be represented by an even number of octal digits. NOTE: Unpredictable results occur when characters are changed to 00 codes. See table 2-2 for a list of valid display codes and their various graphic counterparts.

n = number of lines (starting at the current pointer position) that should be affected by the specified conversion if they contain at least one occurrence of oct1. Highest allowable value = 99999; default value = 1. When users want to convert every appropriate line between the current pointer position and END OF FILE, they enter an * for this parameter. If n = 0, XEDIT will not advance the pointer, and any changes will occur at the current line pointed to.

Example

```
NEW,PASS
READY.
ASCII
READY.
TEXT
ENTER TEXT MODE.

TODAY'S PASSWORD IS:
ORANGE

EXIT TEXT MODE
PACK
READY.
XEDIT
  XEDIT 3.0.0
?? PRINT*
TODAY'S PASSWORD IS:
ORANGE
  END OF FILE
?? NEXT
ORANGE
?? XADD
? $$WWWWW$MMMMMM
?? ↑PRINT*
TODAY'S PASSWORD IS:
ORANGE$$WWWWW$MMMMMM
  END OF FILE
?? OCTCHANGE 53 7655
ORANGE
?? ↑PRINT*
TODAY'S PASSWORD IS:
ORANGE
  END OF FILE
??
```

Example Explanation: After listing the file, the user issues a NEXT command to position the pointer at the second line in the file. Then, he issues an ADD command (prefixed with an X to suppress verification). In response to the single ?, user indicates he wants to add the string \$\$\$\$ to the end of the line designated by the current pointer position. User then issues a PRINT command to see how the ADD command affected the file. Next, the user issues an OCTCHANGE command to have a carriage return (octal code 7655) replace each instance of dollar sign (octal code 53). After XEDIT verifies the single line that was modified, the user issues another PRINT command to see how the entire file was altered.

TABLE 2-2. DISPLAY CODE CONVENTIONS[†]

Display Code	Graphic Representation	Display Code	Graphic Representation
00	:	31	Y
01	A	32	Z
02	B	33	0(zero)
03	C	34	1
04	D	35	2
05	E	36	3
06	F	37	4
07	G	40	5
10	H	41	6
11	I	42	7
12	J	43	8
13	K	44	9
14	L	45	+
15	M	46	-
16	N	47	*
17	O	50	/
20	P	51	(
21	Q	52)
22	R	53	\$
23	S	54	=
24	T	55	(Space)
25	U	56	,
26	V	57	.
27	W	60	#
30	X	61	[

[†]For full ASCII character set, see appendix B of the CYBERNET Interactive Service Time-Sharing Usage Manual listed in the preface.

TABLE 2-2. DISPLAY CODE CONVENTIONS (Cont'd)[†]

Display Code	Graphic Representation	Display Code	Graphic Representation
62]	71	?
63	%	72	<
64	"	73	>
65	-	74	@
66	!	75	\
67	&	76	↑
70	'	77	;

[†]For full ASCII character set, see appendix B of the CYBERNET Interactive Service Time-Sharing Usage Manual listed in the preface.

MANIPULATING FILES

Users can manipulate files during editing by employing the following XEDIT commands:

- COPY -- to copy a particular set of lines from the edited file and write them onto another file while keeping the edited file intact.
- COPYD -- to copy a particular set of lines from the edited file and write them onto another file. However, in this instance, the copied lines are deleted from the edited file.
- READ -- to copy the contents of specified local files onto the edited file.
- READP -- to copy the contents of specified permanent files onto the edited file.

COPYING SPECIFIED LINES ONTO ANOTHER FILE (COPY AND COPYD COMMANDS)

When users want to copy lines from the edited file and write them onto a separate working (local) file, they can issue COPY or COPYD commands. Both of these commands perform the same function, except that the edited file remains in its original form when COPY commands are issued. However, the copied lines are deleted from the edited file with COPYD commands. The following rules apply to both COPY and COPYD commands.

- The copy process begins at the current pointer position and will include all lines in the file that fall between the current pointer position and the position identified by the nth occurrence of a specified string. If XEDIT reaches END OF FILE while executing this command, all lines in the file between the initial pointer position and END OF FILE are copied to the working file.

- After the copy process is completed, the pointer is positioned at the last line which is copied, or top if END OF FILE is encountered.
- When verify mode is in effect, XEDIT only prints those lines that contain the specified string.
- Before the first copying procedure begins, XEDIT causes the working file to be rewound. Consecutive copies onto the same working file result in the copied information begin added to the end of that working file.
- If the user wants to restrict what columns the specified string should occur in before the line is copied, the windowing feature can be used with the COPY and COPYD commands (see WMARGIN command).

The following formats apply to COPY commands:

<u>Format</u>	<u>Function</u>
COPY <u>fname</u> <u>n</u> (CR) or	Copies <u>n</u> lines from the edit file onto file <u>fname</u> ; the edit file remains intact.
COPY <u>fname</u> / <u>string</u> / <u>n</u> (CR) or	Copies inclusively all lines from the edit file current pointer position to file <u>fname</u> until either the <u>string</u> line count <u>n</u> is satisfied or END OF FILE is encountered.
COPY <u>fname</u> / <u>string1</u> .. <u>string2</u> / <u>n</u> (CR) or	Same as above except <u>string</u> may be specified as two strings (<u>string1</u> and <u>string2</u>) that are separated by an indeterminate number of other characters.
COPY <u>fname</u> / <u>string1</u> --- <u>string2</u> / <u>n</u> (CR) or	Same as above except that the string count <u>n</u> is decremented only when a line contains <u>string1</u> which is not followed by <u>string2</u> .
COPY <u>fname</u> /--- <u>string2</u> / <u>n</u> (CR)	Same as above except that the string count <u>n</u> is decremented only if the line does not contain <u>string2</u> .

where:

fname = name of the working file onto which the copied lines will be written; fname can be one to seven alphanumeric characters in length. If fname = OUTPUT, the specified lines are copied to OUTPUT. This is useful for context printing. If fname = NULL and COPYD is used, the specified lines are deleted. This is useful for context deletions. The NULL specification can also be used with COPY.

NOTE

At least one blank must appear before and after the fname entry, otherwise an error diagnostic will be generated.

n = line or string-line count; number of lines to be copied or number of lines which contain at least one occurrence of the specified string that XEDIT must locate before terminating its copying process. Highest allowable value = 99999; default value = 1. When users want to copy every line between the current pointer position and END OF FILE, an * should be entered for the n parameter and the string parameter should be left out. If n = 0 and the string is not found, the pointer position remains the same and the string is not copied; however, the working file fname is rewound if this is the first COPY specified.

string = string of alphanumeric characters for which XEDIT will search when attempting to terminate the copy process.

Example

```
?? LOCATE/*DIVISION/
*DIVISION 1*
?? COPY DIV1 /*END/
*END OF DIVISION*
?? END,,RL
ADDRESS REPLACED
ADDRESS IS A LOCAL FILE

READY.
LNH,F=DIV1
*DIVISION 1*
00130 A.B. NEWTON
00140 166 HASKELL CIRCLE ZIP 55713
00150 EXT 227
00157 A.P. MACDONALD
00158 1313 LEMONTREE AVE ZIP 55722
00159 EXT 5339
00160 Q.E. SMITH
00170 18 PARK PLACE APT 111 ZIP 55704
00180 EXT 8866
*END OF DIVISION*
```

Explanation: Initially, user sets pointer to the line which reads *DIVISION 1*. Then, the user issues a COPY command to copy all entries that fall between the current pointer and the first string which contains *END onto a local file called DIV1. After issuing an END command to terminate XEDIT processing (RL parameter is employed), the user issues a NOS LNH command to review what information was copied onto DIV1.

COPYD commands employ the same parameters as COPY commands. Again, the only difference between these two commands is that the extracted lines are deleted from the edited file when a COPYD command is issued. COPYD commands take the following forms:

Format

Function

```
COPYD fname n (CR)
or
COPYD fname /string/n (CR)
or
COPYD fname /string1..string2/n (CR)
or
COPYD fname /string1---string2/n (CR)
or
COPYD fname /---string2/n (CR)
```

Performs the same function as COPY except the copied lines are deleted from the edit file.

CAUTION

When COPY or COPYD is issued using the string parameter, all lines in the file between the current pointer position and until END OF FILE or the n parameter string-line count is satisfied will be copied to the file fname. Thus COPY and COPYD do not merely extract just lines containing the specified string.

MERGING FILES INTO THE EDITED FILE (READ AND READP COMMANDS)

When users want to merge one or more files into the edited file, they employ READ or READP commands. While both commands function in exactly the same manner, READ commands apply when the user wants to copy local files. Alternatively, READP commands pertain to copying permanent files and, thus, eliminate the need for the user to issue NOS, GET, ATTACH, or OLD commands. The following general rules govern use of both READ and READP commands.

- The specified local or permanent file is copied onto the edited file after the current pointer position, starting with the first record, and continuing until either an empty record or until the first --EOF-- mark or END OF FILE mark is sensed. Embedded --EOR-- marks are maintained.
- Once each copying operation is completed, the pointer is positioned at the last line that was copied onto the edited file.
- XEDIT rewinds the specified local or permanent files before and after the copying operation.
- If a specified local or permanent file can not be read (for example, it does not exist or the user entered the wrong file name), XEDIT immediately terminates its execution of the READ or READP command without attempting to copy any more specified files.

READP commands employ the following format:

Format

READP fname1 fname2...fnamen (CR)

where:

fname1 = name of the first file that the user wants copied onto the edited file immediately after the current pointer position.

fname2 = name of the second file that should be copied (that is, if the user wants to copy more than one file).

fnamen = name of subsequent files that the user wants copied in the sequential order in which they should appear on the edited file.

Example

```
?? P*
*EXECUTIVE MANAGEMENT*
00100 A.B. KRAMER           HQR24
00110 J.J. THOMPSON        SWP19
00120 B.C. MILLER          HQR44
END OF FILE
?? 120
00120 B.C. MILLER          HQR44
?? READP DIVM1 RAD1
?? ↑PRINT*
*EXECUTIVE MANAGEMENT*
00100 A.B. KRAMER           HQR24
00110 J.J. THOMPSON        SWP19
00120 B.C. MILLER          HQR44
*DIVISIONAL MANAGEMENT--PRT DIVISION*
00100 F.R. OLSON           RPT01
00110 M.L. MORRIS          RPT23
00120 B.V. ELLIOT          RPT24
00130 T.C. ROWE            RPT48
00140 H.K. MCGUIRE         RPT89
```



```

*RPT RESEARCH AND DEVELOPMENT*
00100 E.R. STANLEY          RPT55
00110 R.V. HOIM            RPT56
00120 W.D. ALTHOLZ        RPT57
00130 K.B. BELLMON        RPT58
END OF FILE
??

```

Example Explanation: After listing the file by issuing an abbreviated PRINT command, the user sets the pointer to line 120. Then, he issues a READP command to copy the DIVM1 and RAD1 files into the edited file. Subsequently, the user issues a PRINT command with a + prefix to list the modified file to see how the READP command affected its contents.

READ commands follow the same format and parameters as READP commands. However, as noted earlier, READ commands copy each local file while READP commands pertain to permanent files. READ commands can also be applied to files that were created during an XEDIT operation (for example, via COPY and COPYD commands). This is useful when the user needs to move text from one part of the file to another. A local file to be read by the READ command could also have been created by NOS commands such as GET, ATTACH, OLD, NEW, or LIB.

READ fname1 fname2...fnamen (CR)

GENERALIZED COMMANDS

This section discusses a series of generalized XEDIT commands that apply to the following functions:

- EXPLAIN command -- to request a more detailed explanation of an XEDIT message.
- HELP command -- to request information about specific XEDIT commands.
- NOBELLS command -- to tell XEDIT to not ring the user's terminal bell.
- RESTORE command -- to cancel all editing operations performed since the last time the pointer was positioned to the beginning-of-file.
- TEOF and TEOR commands -- to toggle the printing of the messages --EOF-- and --EOR-- respectively.
- WHERE command -- to print the current line count.
- Repeating commands -- to advance the pointer and reexecute the last command that was entered.

REQUESTING DETAILED MESSAGE EXPLANATIONS (EXPLAIN COMMAND)

EXPLAIN commands let users request more detailed information about any XEDIT message, such as an error message that has just printed out or the most recent message that was printed. The messages that apply to the EXPLAIN command are listed in appendix A. To get a message explained in detail, the user uses the following form:

<u>Format</u>	<u>Example</u>
EXPLAIN (CR)	?? GET,PLAN NO SUCH COMMAND ?? EXPLAIN EXPLANATION OF- NO SUCH COMMAND THE COMMAND IS ILLEGAL OR AN IMPROPER SEPARATOR WAS USED AFTER THE COMMAND.

Example Explanation: User enters an illegal command and issues an EXPLAIN command to request an explanation of the message "NO SUCH COMMAND."

REQUESTING INSTRUCTIONAL INFORMATION (HELP COMMAND)

HELP commands let users request information about each of the various XEDIT commands. Accordingly, this instructional information can be requested in the following form:

<u>Format</u>	<u>Example</u>
HELP, <u>cmd</u> (CR) or H, <u>cmd</u> (CR)	?? HELP,PRINT PRINT \$ [P] ACTION: PRINT \$ LINES STARTING AT THE POINTER. THE POINTER IS POSITIONED AFTER THE LAST LINE PRINTED. ??

where:

cmd = any valid XEDIT command or its abbreviation. Default: If the user does not enter any command, XEDIT explains how the HELP command is used and lists all XEDIT commands.

Explanation: User issues HELP command to request information about the PRINT command and XEDIT responds.

SUPPRESSING TERMINAL BELL RINGING (NOBELLS COMMAND)

The NOBELLS command allows the user to turn off the bell that rings when various XEDIT messages print out. XEDIT defaults to ringing the bell unless the NOBELLS command is issued. The form is as follows:

<u>Format</u>	<u>Example</u>
NOBELLS (CR) or NB (CR)	?? NOBELLS ?? PRINT/ ILLEGAL PARAMETER ??

Example Explanation: User issues a NOBELLS command to prevent the bell from ringing when subsequent messages are printed. The user tests to see if the bell has been locked from ringing by issuing a PRINT command with an illegal parameter to get the error message ILLEGAL PARAMETER which normally rings the bell.

CANCELING EDITING OPERATIONS (RESTORE COMMAND)

Users issue RESTORE commands to cancel all the editing operations that have been performed since the last time the pointer was positioned to the beginning-of-file. Typically, this command is entered when the user determines that one (or more) recently issued command was not correctly entered or did not accomplish what the user originally intended. Changes made before the following conditions can not be restored to their previous state:

1. an END OF FILE message is issued;
2. a TOP command was executed;
3. an up-arrow (↑) command prefix was executed;
4. a NEXT -n command was executed; or
5. a find line number (ln) command was executed so that a circular (wrap-around) operation was necessary.

<u>Format</u>	<u>Example</u>
RESTORE (CR)	?? XLOCATE/SMITH/ ?? DELETE 3 00160 Q.E. SMITH 00170 18 PARK PLACE APT 111 ZIP 55704 00180 EXT 8866 ?? RESTORE ?? 160 00160 Q.E. SMITH ??
or	
REST (CR)	

Example Explanation: User sets pointer to first line which contains the string SMITH. However, since the LOCATE command was prefixed by an X, no verification of the current pointer position occurs. User then issues a DELETE command to delete the next three lines. XEDIT responds by verifying what lines were deleted. At the time, the user realizes he deleted information about the wrong SMITH. Consequently, he issues a RESTORE command to cancel the preceding DELETE instruction. To verify that the information was restored, the user attempts to locate line 160. XEDIT indicates that line 160 was, in fact, restored.

SUPPRESSING THE PRINTING OF FILE MARKS (TEOF AND TEOR COMMANDS)

TEOF and TEOR commands cause XEDIT to alternate or toggle between printing and no printing of --EOF-- (end-of-file) messages and --EOR-- (end-of-record) messages. By default, XEDIT is in a state of printing these file mark messages. To perform this toggle, the user issues commands of the following form:

<u>Format</u>	<u>Example</u>
TEOF (CR)	?? PRINT*
TEOR (CR)	*EXECUTIVE MANAGEMENT*
	--EOR--
	00100 A.B. KRAMER HQR24
	00110 J.J. THOMPSON SWP19
	--EOR--
	--EOF--
	00120 B.C. MILLER HQR44
	END OF FILE
	?? TEOF
	?? TEOR
	?? PRINT*
	EXECUTIVE MANAGEMENT
	00100 A.B. KRAMER HQR24
	00110 J.J. THOMPSON SWP19
	00120 B.C. MILLER HQR44
	END OF FILE
	??

Example Explanation: After listing the file by issuing a PRINT command, the user notices that there are file marks --EOR-- and --EOF-- in the file. The user decides that the file marks should exist in the file, but does not want them printed on the terminal so the TEOF and TEOR commands are issued. The user then issues another PRINT command to list the file again as a verification that the --EOR-- and --EOF-- messages are not printed.

If the user wants to actually delete file marks, rather than merely suppress the printing of them, see the DEOF and DEOR commands.

Four additional forms of TEOF and TEOR commands exist:

TEOF + (CR)	to turn on the printing of --EOF-- messages
TEOF - (CR)	to turn off the printing of --EOF-- messages
TEOR + (CR)	to turn on the printing of --EOR-- messages
TEOR - (CR)	to turn off the printing of --EOR-- messages

PRINTING THE CURRENT LINE COUNT (WHERE COMMAND)

WHERE commands let a user determine the "current line count." Calculated and printed by the editor, this number indicates how many file lines appear between the beginning-of-file and the current pointer position.

<u>Format</u>	<u>Example</u>
WHERE (CR)	?? 157
or	00157 A.P. MACDONALD
W (CR)	?? WHERE
	9
	??

Example Explanation: After setting the pointer to line 157, the user issues a WHERE command to determine how far (that is, in terms of number of lines) line 157 is from the first line in the file. XEDIT responds by indicating the pointer is now positioned at the ninth line in the file.

REPEATING COMMANDS

Users can repeat the execution of the preceding command at a different pointer position by entering a period followed by an n parameter. In this instance, the n parameter indicates how much further the pointer should be advanced before the preceding command is reexecuted. Additionally, users can advance the pointer and reexecute a preceding Z or Y command by entering a minus sign followed by an n parameter.

Format

.n (CR)

or

-n (CR)

Example

```
?? PRINT
00130 A.B. NEWTON
?? .3
00157 A.P. MACDONALD
?? .3
00160 Q.E. SMITH
??
```

where:

. = reexecute the preceding normal XEDIT command.

- = reexecute the preceding Z or Y command.

n = number of lines that the pointer should be advanced before re-executing the preceding command. Default value = 1. A value = 0 means execute the last command without advancing the pointer.

Explanation: User issues a PRINT command to list the line designated by the current pointer position. XEDIT responds by printing line 130. Subsequently, the user enters .3 to specify that XEDIT should advance the pointer three lines and reexecute the PRINT command. XEDIT responds by printing line 157.

Finally, the user issues another .3 entry to advance the pointer another three lines and reexecute the PRINT command. XEDIT reacts by listing line 160.

SUBMITTING MULTIPLE ENTRIES IN A SINGLE LINE

XEDIT users can issue more than one entry in a single line by issuing any of the following command variations:

- DELIMIT commands and delimited command sequences -- to enter more than one command in a single line separated by delimiters, including editing data (if + prefix is used). Delimited command sequences enable the user to reduce the amount of time that the user is connected to the terminal and thereby reduce costs.
- Z or Y commands -- to enter several XEDIT commands in a single line, including editing data (if + prefix is used). The primary use of a Z or Y command sequence is to enable the user to repeat the sequence at a later time by using the -n command (see above).
- + prefix commands within delimited command sequences or Z or Y commands -- to tell XEDIT that, for the specified command, editing data will appear as the next delimited item on the same line as the command itself, instead of the normal primary input source (single question mark (?)).

ENTERING MULTIPLE COMMANDS AND DATA (DELIMIT COMMAND)

Users can enter both multiple commands and editing data in a single line by first issuing a DELIMIT command and then entering several commands and data on a subsequent line. The basic rules governing this procedure include:

- Any valid XEDIT command can appear within the user's single line of delimited commands and editing data.
- The user employs the delimiter specified in the DELIMIT command to separate commands and editing data.
- The delimit character specified in the DELIMIT command stays in effect until the user issues a subsequent DELIMIT command.
- The delimiter used in a Y or Z command must not be the same character as the delimiter specified by the current DELIMIT command execution.
- Both commands and editing data (for example, input related to an ADD command) can appear within a delimited command sequence. However, if a command requires editing data and the user wishes to include it on the same line as the command, the user should use the + prefix character (see pages 2-4 and 2-59). Otherwise, editing data is requested as needed via a single question mark prompt.

DELIMIT commands should be issued in the following form:

Format

DELIMIT char (CR)

or

DEL char (CR)

where:

char = any numeric or special character (except a space or alphabetic) which the user wants to establish as the delimit character. This character will be used to separate the several commands or editing data that appear in a subsequent line of user entries. Default: If the user issues a DELIMIT command and omits the char parameter, XEDIT assumes the user wants to clear the effect of an earlier DELIMIT command. To specify a comma use: DELIMIT,,

Example

```
?? PRINT 2
00220 B.P. COLLINS
00230 710 ELM ST
?? DELIMIT &
?? +XA& ZIP 55722&+I&00240 EXT 726&22OP3
00220 B.P. COLLINS
00230 710 ELM ST ZIP 55722
00240 EXT 726
??
```

Explanation: User issues PRINT command to list two lines. After XEDIT responds with the result that the point is positioned to line 230, the user issues a DELIMIT command to establish an ampersand (&) as the line delimiter. Then, the user issues a single line that contains both commands and editing data. This multiple input is separated by numerous user-selected line delimiters. Specifically, this line contains: 1) an abbreviated + and X prefixed ADD command, 2) the editing data that accompanies the ADD command, 3) an abbreviated + prefixed INSERT command, 4) the editing data that accompanies the INSERT command, and 5) an abbreviated PRINT command to list the portion of the file that was modified by this series of commands, beginning at line 220. Note the use of the plus prefix to enable XEDIT to read editing data input from the command list instead of normal input.

A command sequence using the DELIMIT character should be issued in the following form:

cmd1 char cmd2 char ... char cmdn

where:

cmd = a legal XEDIT command, command data or directives (if plus (+) prefix is used) which the user wants to process. XEDIT will execute each command according to its sequential (left to right) position in the list of component commands. The command or command parameters must not contain the DELIMIT character char.

char = any legal character as specified by the DELIMIT command above.

If any command encounters an error, such as ILLEGAL PARAMETER, all remaining commands in the list are ignored and the editor requests the next command line.

Users can tell XEDIT that command data or command directives will appear as the next delimited item on the same line as the command itself by making use of the + prefix character (see pages 2-4 and 2-59). Otherwise, the editing data is requested as needed by XEDIT via a single question mark (?) prompt.

ENTERING MULTIPLE COMMANDS AND DATA (Z AND Y COMMANDS)

Z and Y commands enable users to issue more than one command including editing data (if plus (+) prefix is used) in a single line. In these instances, the user enters the letter Z or Y, followed by a list of XEDIT commands. Each command in the list must be separated from the other commands by means of a user-selected delimiter. While Z commands perform the same function as Y commands, they differ in their procedural operations. When a Z command is executed, XEDIT prints each component command as it is processed by the editor. However, Y commands are executed without listing the component command as it is being processed.

The following basic rules apply to both Z and Y commands:

- Users can issue any valid XEDIT command within their list of component commands except Z and Y commands. If a DELIMIT command is issued in a Z or Y command, the DELIMIT character takes effect only after XEDIT has executed all component commands in the Z or Y command sequence. In other words, a DELIMIT command character defined within a Y or Z command sequence does not affect the Y or Z command delimiter. However, the Y or Z command delimiter should never be the same as a delimiter specified by a DELIMIT command.
- If any command encounters an error, such as ILLEGAL PARAMETER, all remaining commands in the list are ignored and the editor requests the next command line. This action prevents XEDIT from accidentally ruining the file with the remaining commands and also allows the user to use the EXPLAIN command to find out what caused the error.
- Both commands and editing data (for example, input related to an ADD command) can appear within a Z command. However, if editing data is included, the + prefix character must be included with the command (see pages 2-4 and 2-59).
- Users can employ any alphanumeric or special character as the command delimiter so long as that character does not appear within any of the component commands or their command strings, and provided it is not the same as the DELIMIT command delimiter.

NOTE

It is highly recommended that users employ dollar signs (\$) or semi-colons (;) as command delimiters, while slashes (/) are suggested for string delimiters.

- Z commands cause XEDIT to list each component command as it is executed. Y commands suppress this printing.
- Z and Y commands can be reexecuted at an advanced pointer position by entering -n (see page 2-55).

Z and Y commands should be issued in the following form:

Format

Z\$cmd1\$cmd2\$...\$cmdn (CR)

or

Y\$cmd1\$cmd2\$...\$cmdn (CR)

where:

\$ = command delimiter. This must be different from the DELIMIT command delimiter.

cmd = a legal XEDIT command, command data, or command directives the user wants to process; XEDIT will execute each command according to its sequential (left to right) position in the list of component commands

Example

```
?? Z$LOCATE/SMITH/$+ADD$ JR.$DELETE/BEE/
?? LOCATE/SMITH/
00160 Q.E. SMITH
?? +ADD
00160 Q.E. SMITH JR.
?? DELETE/BEE/
00190 P.T. BEE
??
```

Explanation: User issues a Z command that contains three component commands (LOCATE, ADD, and DELETE); in this instance, the user employs a dollar sign as the command delimiter. As XEDIT executes each component command, it prints the command and any appropriate verification. First, XEDIT indicates the LOCATE command has been executed; verification is printed. Since editing data is required, the user issues a plus (+) prefix character on the ADD command. This tells XEDIT to take the next delimited item as the input string which, in this case, is the word JR. This is added to the end of line 160. XEDIT proceeds to verify this instruction. Finally, the DELETE command is executed and verified.

ENTERING MULTIPLE COMMANDS AND DATA (+ PREFIX)

Z and Y commands and delimited command sequences normally assume that editing data (that is, input related to the ADD, INSERTB, MODIFY, QMOD, REPLACE, and YQMOD commands) will be specified by the user from the normal input source (a single question mark (?)). However, users can tell XEDIT that editing data will appear as the next delimited item on the same line as the command itself by making use of the plus (+) prefix character. The format of a command using the plus prefix character is given in the General XEDIT Conventions section on page 2-4 and as follows:

<u>Format</u>	<u>Example</u>
<u>+ cmd</u> (CR)	?? DEL \$
where:	?? L/ADDRESS:/\$+INSERT\$ZIP CODE:
<u>cmd</u> = one of the following XEDIT commands:	ADDRESS:
ADD QMOD	?? +PRINT*
INSERT REPLACE	NAME:
INSERTB YQMOD	ADDRESS:
MODIFY	ZIP CODE:
	END OF FILE
	?? ADD\$+/ADD\$POB 55\$+/ADD\$55703
	? Q.E. SMITH JR
	NAME: Q.E. SMITH JR
	ADDRESS: POB 55
	ZIP CODE: 55703
	?? +PRINT*
	NAME: Q.E. SMITH JR
	ADDRESS: POB 55
	ZIP CODE: 55703
	END OF FILE
	??

Example Explanation: The user issues an abbreviated DELIMIT command to establish a dollar sign (\$) as the line delimiter. Then, the user issues a delimited command sequence that contains both commands and editing data. Specifically, this line contains: 1) an abbreviated LOCATE command, 2) an INSERT command with the plus (+) prefix, and 3) one line of editing data that accompanies the INSERT command. The user then issues a +PRINT* to list the entire file to verify that the line was properly inserted. Next, the user enters a delimited command sequence (commands separated by the dollar sign (\$) delimiter character). The first command in the sequence is an ADD command without a plus (+) prefix character. Since the plus (+) prefix is not used, XEDIT requests the user to enter a line of editing data such as Q.E. SMITH JR from the normal input source (single question mark (?)). Next, the plus (+) and slash (/) prefix characters are issued on two more ADD commands. The editing data POB 55 and 55703 are added from the same line as the command itself since the plus prefix was specified. The slash tells XEDIT to skip to the next line before executing the command. Finally, the user issues another +PRINT* to verify the ADD operation on all lines of the file.

TAB CONTROL

Users can control tab settings during editing by employing the following XEDIT commands:

- DEFTAB -- to define a tab character for subsequent use in entering editing data with the INSERT, INSERTB, REPLACE, INPUT and **CR** commands.
- LISTAB -- to list the current tab character and tab stop positions.
- TABS -- to define tab stop column positions.

DEFINING THE TAB CHARACTER (DEFTAB COMMAND)

Users can define a tab character by issuing a DEFTAB command in the following form:

<u>Format</u>	<u>Example</u>
DEFTAB <u>char</u> CR	?? DEFTAB \$
or	?? INSERT
DT <u>char</u> CR	? \$MTD\$YTD\$TOTAL
	?? PRINT
	MTD YTD TOTAL

where:

char = the tab character to be used in entering editing data with the INSERT, INSERTB, REPLACE and **CR** command. Default: If the user issues a DEFTAB command and omits the char parameter, XEDIT assumes the user wants to clear the effect of an earlier DEFTAB command. In order to avoid confusion on input, there is no default tab character. To specify a comma use: DEFTAB,,

Explanation: User issues a DEFTAB command to establish dollar sign (\$) as the tab character. Using the tab character with the INSERT command edit data, the column headings "MTD", "YTD", and "TOTAL" are created spaced by the default tab column positions (11 18 30). The user then enters a PRINT command to verify the tabbing operation.

DEFINING THE TAB POSITIONS (TABS COMMAND)

Users can define up to eight tab column positions by issuing a TABS command in the following form:

Format

TABS t_1 t_2 t_3 ... t_8 (CR)

or

TAB t_1 t_2 t_3 ... t_8 (CR)

where:

t = tab column positions with increasing (left to right) values between 1 and 160 inclusive; default values are 11 18 and 30 (COMPASS Assembly Language). If the user issues a TABS command and omits the t parameters, XEDIT assumes the user wants to clear the effect of an earlier TABS command. Any tab stops not defined have no effect on the tabbing and any tab characters given in the input line which occur past the last tab stop will be copied to the file.

Example

```
?? DEFTAB;
?? TABS 5 10 15 20
?? INSERT 7
? I. STATISTICS
? ;A. PROBABILITY
? ;B. EXPECTED VALUE
? ;; 1. AVERAGE
? ;; 2. VARIANCE
? ;; 3. STANDARD DEVIATION
? ;C. REGRESSION THEORY
?? ^PRINT*
I. STATISTICS
  A. PROBABILITY
  B. EXPECTED VALUE
    1. AVERAGE
    2. VARIANCE
    3. STANDARD DEVIATION
  C. REGRESSION THEORY
END OF FILE
??
```

Example Explanation: User issues a DEFTAB command to establish semi-colon (;) as the tab character. The user then issues a TABS command to set the tab column positions at 5, 10, 15, and 20. The user issues an INSERT command and begins inserting lines and the tab character where appropriate tabbing is desired. Finally, the user enters an ^PRINT* command to verify that the tabbing was correctly performed.

NOTE

Tab positions 15 and 20, although set, were not used in this example.

LISTING THE TAB SETTINGS (LISTAB COMMAND)

Users can list the current tab character and tab stop column positions by issuing a LISTAB command in the following form:

Format

LISTAB (CR)

or

LT (CR)

Example

```
?? DEFTAB;
?? TABS 4 8 12
?? LISTAB
; TABS 4 8 12
??
```

Example Explanation: User issues a DEFTAB command to establish semi-colon (;) as the tab character. The user then issues a TABS command to set the tab column positions at 4, 8, and 12. Finally, the user issues a LISTAB command to list the current tab character and tab stop column positions.

MARGIN AND TRUNCATION CONTROL

During XEDIT processing, the editor operates under control of a maximum line width of 160 characters which defines how many characters can be entered per editing line. If the user enters more than 160 characters, XEDIT automatically truncates the line and sends the following message:

TRUNCATION AT n

where:

n = number of lines from the beginning of the file where the truncation occurred.

If the user desires more control over truncation and the right margin during editing, the following commands can be used:

- RMARGIN -- to set the right margin position.
- FINDLL -- to find lines longer than the current right margin RMARGIN setting.
- TRUNC -- to truncate long lines as defined by the RMARGIN and the FINDLL commands.

SETTING THE MARGIN (RMARGIN COMMAND)

The RMARGIN command is used for setting the column position of the right margin of a file. Lines longer than this value are considered to be "long lines" by the find long line FINDLL command and are truncated to the right of this position by the TRUNC command. After this command is executed, the pointer retains its original position.

Format

RMARGIN m (CR)

or

RM m (CR)

Example

?? RMARGIN 140
??

Explanation: User issues a RMARGIN command to specify that subsequent FINDLL and TRUNC commands base their operations on a right margin of 140 characters per line.

where:

m = column position of the right margin of the file. Range of allowable entries is 10 to 160.

NOTE

This command only has effect when used in conjunction with the FINDLL and TRUNC commands.

FINDING LONG LINES (FINDLL COMMAND)

When users want to locate a specific number of long lines, they can issue FINDLL commands. Long lines are defined as lines which are longer than the current right margin setting (see RMARGIN command). This command is most useful when the user is operating in verify mode since the long lines are listed in this situation. The following command format is valid for entering FINDLL commands:

Format

FINDLL n (CR)

or

FLL n (CR)

where:

n = number of lines that are longer than the current RMARGIN setting which the user wants to locate. An asterisk (*) should be used when the user wants to locate every long line in the file. Highest allowable value = 99999; default value = 1. The file pointer will be positioned at the last long line encountered unless END OF FILE is encountered. A value = 0 assumes n = 1.

Example

```
?? PRINT*
*EXECUTIVE MANAGEMENT*
EX100 S.W. KRAMER           HQR24
EX130 J.J. JOHNSON
EX120 R.E. MILLER          HQR44
EX125 F.R. SCOTT
END OF FILE
?? RMARGIN 25
?? FINDLL*
EX100 S.W. KRAMER           HQR24
EX120 R.E. MILLER          HQR44
END OF FILE
??
```

Explanation: A listing is obtained to determine what right margin setting is needed in order to locate lines which have fields to the right of the name field (for example, S.W. KRAMER). The user decides to get a listing of all lines longer than 25 characters.

TRUNCATING LONG LINES (TRUNC COMMAND)

As mentioned earlier in this section, XEDIT automatically truncates lines greater than 160 characters. If the user desires more control over what lines get truncated within the 160 character limit, the TRUNCATE command in conjunction with the RMARGIN and FINDLL commands provide this capability. The TRUNC command takes the following form:

Format

TRUNCATE n (CR)

or

TRUNC n (CR)

Example

```
?? PRINT*
*EXECUTIVE MANAGEMENT*
EX100 S.W. KRAMER           HQR24
EX130 J.J. JOHNSON
EX120 R.E. MILLER          HQR44
EX125 F.R. SCOTT
END OF FILE
```

where:

n = number of lines that are longer than the current RMARGIN setting which the user wants to truncate. An asterisk (*) should be used when the user wants to truncate every long line in the file from the current pointer position. Highest allowable value = 99999; default value = 1. The file pointer will be positioned at the last truncated line unless END OF FILE is encountered. A value = 0 assumes n = 1.

```
?? RMARGIN 25
?? TRUNCATE*
*EXECUTIVE MANAGEMENT*
EX100 S.W. KRAMER
EX130 J.J. JOHNSON
EX120 R.E. MILLER
EX125 F.R. SCOTT
END OF FILE
??
```

Example Explanation: After a listing of the file is obtained, the right margin is set to position 25. The TRUNCATE command is then issued to truncate all lines greater in length than the RMARGIN setting (25).

TERMINATING XEDIT EXECUTION

XEDIT users can terminate the editor's execution in different ways by issuing different commands. These include:

- FILE commands -- to temporarily suspend editing and save the edited file in its intermediate form
- END and QUIT commands -- to permanently terminate editing and save the edited file
- STOP commands -- to permanently terminate editing without saving any of the editing operations

TEMPORARILY SUSPENDING XEDIT (FILE COMMAND)

FILE commands enable users to temporarily suspend XEDIT execution in order to save an intermediate version of the edited file. Once the user issues a FILE command, the following procedure takes place:

- XEDIT temporarily terminates its execution.
- The entire edited file is saved as a NOS permanent file in the manner designated by the user-entered mode parameter, regardless of where the pointer was positioned when the FILE command was issued.
- XEDIT processing is resumed after the FILE command is successfully executed. The pointer is now positioned to the beginning-of-file.

The form for issuing FILE commands is shown below:

Format

FILE, fname, mode (CR)

or

F, fname, mode (CR)

where:

fname = name (1-7 alphanumeric characters long) of the file onto which the editing operations will be written. Default: If no file name is entered, XEDIT assumes the name of the edited file (that is, the file name specified when XEDIT was first called). If the user wants to omit the fname entry, indicate that omission by typing an extra comma. For example:

FILE,,SAVE

mode = manner in which NOS should dispose of the intermediate file. Valid entries for this parameter are:

S or = edited file should be
SAVE saved as a new indirect access permanent file.

R or = edited file should
REPLACE replace an existing (that is, OLD) indirect access permanent file.

L or = edited file should be
LOCAL written onto a local NOS file.[†]

Example

```
?? 120
00120 EXTENSION 6533
?? CHANGE/EXTENSION/EXT/*
00120 EXT 6533
00150 EXT 227
00159 EXT 5339
00180 EXT 8866
00210 EXT 18
END OF FILE
?? FILE,,R
ADDRESS REPLACED
?? 120
00210 EXT 6533
??
```

Explanation: After positioning the pointer at line 120, the user issues a CHANGE command to replace each reference to EXTENSION with EXT. XEDIT then proceeds to verify which file lines were modified by the CHANGE command. Next, the user issues a FILE command to save this editing change without permanently terminating XEDIT execution. In this command, the user omits the fname parameter. Consequently, the original file name specified when XEDIT was called (that is, ADDRESS) will apply to the intermediate file.

In addition, the user selects R as the mode parameter. Accordingly, the ADDRESS permanent file will be replaced so that it will contain the user's editing changes. Additionally, the user's local edited file retains these changes, as is indicated by the user's positioning the pointer to line 120.

[†]LOCAL mode cannot be selected when using direct access files.

C or = edited file changes
COPY should be copied
back to file fname.
C can only be used
with files that can
be written on. This
parameter is default
for direct access
files attached in
write mode (that is,
the user may simply
type FILE).

RL = performs functions
of both the REPLACE
and LOCAL options.

SL = performs functions
of both the SAVE
and LOCAL options.

Default: If a mode parameter is
omitted, XEDIT assumes the user
wants LOCAL mode, unless the
edit file is direct access. For
direct access edit files the
default is COPY mode.

PERMANENTLY TERMINATING XEDIT (END, QUIT, AND STOP COMMANDS)

END and QUIT commands allow a user to permanently terminate XEDIT execution and at the same time dispose of the edited file in a manner that will save its contents. However, STOP commands merely terminate XEDIT without allowing for any disposal of the edited file (that is, the editing changes can never be retrieved).

STOP Command

Users issue STOP commands when they want to permanently terminate XEDIT execution without saving any of the operations performed during the editing session. In other words, the file will remain exactly as it appeared before XEDIT was called into execution, assuming the user has not issued any FILE commands. If the STOP command is issued in batch mode, the first statement after an NOS EXIT card is executed; otherwise, the job is terminated.

These commands take the following form:

<u>Format</u>	<u>Example</u>
STOP (CR)	?? DELETE 3 00190 P.T. BEE 00200 8710 14TH ST ZIP CODE 55713 00210 EXTENSION 18 END OF FILE


```

?? RESTORE
?? 190
END OF FILE
?? STOP
ABORTED

READY.
XEDIT
XEDIT 3.0.0
?? 190
00190 P.T. BEE
??

```

Example Explanation: User issues a DELETE command to delete three lines, beginning at the current pointer position. XEDIT responds by verifying which lines were deleted. Then, the user realizes he inadvertently deleted the wrong lines. To negate this mistake, the user attempts to issue a RESTORE command but since the pointer reached END OF FILE while executing the DELETE command, the RESTORE command can not restore the deleted lines (as indicated by the user's subsequent unsuccessful attempt to locate line 190). Then, the user issues a STOP command to terminate XEDIT execution. As a result, the editing changes are not retained. This is verified as the user recalls XEDIT and locates one of the originally deleted lines (190).

END and QUIT Commands

When users want to permanently terminate XEDIT execution and dispose (for example, save) their edited files, they issue QUIT or END commands.

NOTE

Both END and QUIT perform exactly the same function and employ identical parameters.

The following general rules pertain to these commands:

- Once the command is issued, XEDIT permanently terminates its execution. Then, the edited file (with all the results of the user's editing operations) is written onto a specified file fname parameter), which in turn is disposed of in accordance with a user-specified mode parameter.
- Assume that a user enters a LOCAL option for the mode parameter. In this instance, the edited file can not be the primary NOS file unless the user's fname entry matches the name of the file that originally existed as the primary file when XEDIT was first called into execution.
- Users should enter SAVE and REPLACE options (as mode parameters) with care. SAVE and REPLACE parameters do not write the edited file modifications onto the existing local version of the file. Only permanent files are affected by SAVE and REPLACE commands. See figure 1-7 for an example of this characteristic.

END and QUIT commands should be entered in the following form:

Format

END, fname, mode (CR)

or

E, fname, mode (CR)

QUIT, fname, mode (CR)

or

Q, fname, mode (CR)

where:

fname = name (1-7 alphanumeric characters long) of the file onto which the editing operations will be written. Default: If no file name is entered, XEDIT assumes the name of the edited file (that is, the name specified when XEDIT was first called into execution) should be employed. If the user wants to omit the fname entry, he must indicate that omission by typing an extra comma (for example, END,,R).

mode = manner in which NOS should dispose of the fname file. Valid entries for this parameter are:

S or = edited file should be
SAVE saved as a new indirect
access permanent file.

R or = edited file should re-
REPLACE place an existing (that
is, OLD) indirect
access permanent file.

C or = edited file changes
COPY should be copied back to
file fname. C can only
be used with files that
can be written on. This
parameter is default for
direct access files
attached in write mode
(that is, the user may
simply type END or
QUIT). See appendix B.

Example 1

```
?? 120
00120 EXT 6533
?? CHANGE/EXT/EXTENSION/*
00120 EXTENSION 6533
00150 EXTENSION 227
00159 EXTENSION 5339
00180 EXTENSION 8866
00210 EXTENSION 18
END OF FILE
?? END,,R
ADDRESS REPLACED
```

READY.

```
LNH
### NAMES/ADDRESSES ARE FICTITIOUS ###
00100 M.T. JONES
00110 1544 WILSHIRE ST ZIP CODE 55722
00120 EXT 6533
00130 A.B. NEWTON
*TERMINATED*
```

Explanation: After positioning the pointer at line 120, the user issues a CHANGE command to replace all occurrences of EXT with EXTENSION. Then, XEDIT responds by verifying which lines were modified. Subsequently, the user issues an END command. Since the fname parameter is omitted, XEDIT employs the name specified when XEDIT was called into execution (that is, ADDRESS). Additionally, the user enters an R option for the mode parameter. As a result, the permanent file version of ADDRESS will contain the above editing change. However, since LOCAL mode was not selected, the local edited file will not contain the above editing operation. This is confirmed when the user issues a NOS LNH command and sees that line 120 was not modified.

Example 2

```
?? 120
00120 EXT 6533
?? CHANGE/EXT/EXTENSION/*
00120 EXTENSION 6533
00150 EXTENSION 227
00159 EXTENSION 5339
00180 EXTENSION 8866
00210 EXTENSION 18
END OF FILE
```

L or = edited file should be
LOCAL written onto a local
NOS file.

RL = performs functions of
both REPLACE and
LOCAL modes.

SL = performs function of
both SAVE and LOCAL
modes.

Default: If a mode parameter is
omitted, XEDIT assumes the user
wants the LOCAL mode, unless
the edit file is direct access. For
direct access edit files the default
is COPY mode.

?? END,,RL
ADDRESS REPLACED
ADDRESS IS A LOCAL FILE

READY.
LNH
NAMES/ADDRESSES ARE FICTITIOUS ###
00100 M.T. JONES
00110 1544 WILSHIRE ST ZIP CODE 55722
00120 EXTENSION 6533
00130 A.B. NEWTON
TERMINATED

Explanation: User proceeds exactly as in Example
1 except that he enters an RL parameter in his
END command. Consequently, both the ADDRESS
permanent file and the local edited file will contain
the above editing change. The user confirms this
by issuing an LNH command to list his local file
copy and seeing that line 120 does contain the
editing modification.

XEDIT DIAGNOSTICS AND MESSAGES

A

Table A-1 lists (in alphabetical order) the error diagnostics and informative messages which XEDIT generates in response to user entries.

TABLE A-1. XEDIT DIAGNOSTICS AND MESSAGES

Diagnostic	Message
ABORTED	XEDIT terminated its execution without recording the editing changes performed during the editing session. However, changes written to files via FILE, COPY or COPYD commands during the session remain intact. This message is not issued during in-line edit mode aborts.
ARGUMENT ERROR	<p>User issued a legal command, but incorrectly specified one or more parameters as follows:</p> <ol style="list-style-type: none">1. One string field is needed and it is missing (for example, /<u>string</u>/).2. Two string fields are needed and the second is missing (for example, /<u>string1</u>..<u>string2</u>/ or /<u>string1</u>---<u>string2</u>/).3. While the user was in INPUT mode (used the INPUT command with the escape character), a line count <u>n</u> or string count <u>m</u> parameter value other than zero was used on a command which might possibly move the pointer. Some affected commands are CHANGE, LOCATE, COPY and REPLACE.4. While the user was in INPUT mode (used the INPUT command with the escape character), the up-arrow (↑) or slash (/) prefix characters were used on a command. Pointer movement is not allowed while under INPUT mode.5. Margin specifications on the WMARGIN command are in the wrong order (for example, WM 10 5).6. The Y or Z command list is missing.7. In OCTCHANGE, an even number of digits was not specified for the octal number.

TABLE A-1. XEDIT DIAGNOSTICS AND MESSAGES (Cont'd)

Diagnostic	Message
BAD FILE NAME	<p>User either did not specify a required file name or specified a file name which:</p> <ol style="list-style-type: none"> 1. Contains bad characters or is over seven characters in length. 2. Is reserved by XEDIT.
BAD TEXT LINE ENCOUNTERED	<p>The last word of a nonempty record does not contain an end of line byte. This message most frequently occurs when the user forgets to issue a PACK command after leaving TEXT mode when processing interactively. XEDIT aborts.</p>
BATCH ABORT - COMMAND ERROR	<p>While in batch mode, a command syntax or parameter error (other than DELIMITER) was encountered. XEDIT aborts.</p>
BATCH ABORT - RETRY COUNT EXCEEDED	<p>XEDIT allows only one retry for "NAME EDIT FILE" if the user has a mass storage input or output file. XEDIT aborts.</p>
CANNOT EDIT EXECUTE ONLY FILES	<p>User attempted to edit a file which has been stored in a user's permanent file area by using the M=E parameter on the NOS SAVE or DEFINE control cards. This is a security feature of the NOS system.</p>
COMMAND NOT VALID	<p>User issued a legal command, but not valid for use under INPUT mode or CREATION mode as follows:</p> <ol style="list-style-type: none"> 1. If under INPUT mode, user can not issue a command which will <u>always</u> move the pointer (for example, DELETE, COPYD, FILE, etc.). 2. If under CREATION mode, user can not issue a command which assumes the presence of an existing text line (for example, DELETE, INSERTB).
COMMAND STACKING ERROR	<p>User attempted to recursively call an input source (for example, using a Y or Z command within another Y or Z command such as Y/Y;P/Y;P;N).</p>

TABLE A-1. XEDIT DIAGNOSTICS AND MESSAGES (Cont'd)

Diagnostic	Message
DE LIMITER	User omitted the closing delimiter of a delimited string parameter. As this is a warning message only, XEDIT will execute the command by assuming that a string delimiter should appear after the last nonblank character of the string (for example, L/AB CDE will be interpreted as L/AB CDE/).
EDIT	User pressed the carriage return or entered the <u>e</u> EDIT command to terminate the line input prompts that are produced while under INPUT mode. XEDIT will now accept normal editing commands.
EMPTY FILE/CREATION MODE ASSUMED	User entered XEDIT without specifying a file to edit or by specifying a file that contains no information. XEDIT automatically puts the user in CREATION mode to create a new edit file (see C parameter on the XEDIT control statement in appendix C).
END OF FILE	XEDIT read the end-of-information mark while executing the most recent command. XEDIT will no longer continue to execute that command and will position the file pointer to the beginning-of-file.
END OF INPUT ENCOUNTERED - ABORTED	While in batch mode an end-of-record, end-of-file, or end-of-information mark was encountered on the input file. XEDIT aborts.
---EOF---	XEDIT read an end-of-file mark. Execution of the command will continue until its normal termination. The end-of-file mark will be retained in the user's file unless a DEOF command is being executed. This message can be turned off by use of the TEOF command.
---EOR---	XEDIT read an end-of-record mark. Execution of the command will continue until its normal termination. The end-of-record mark will be retained in the user's file unless a DEOR command is being executed. This message can be turned off by use of the TEOR command.
ERROR IN XEDIT ARGUMENTS	User issued an XEDIT control card with illegal options specified (see appendix A on Batch Command Processing). This most commonly occurs when the user issues the I= parameter with the Q NOS command or when the Q NOS command is issued with no in-line edit commands.

TABLE A-1. XEDIT DIAGNOSTICS AND MESSAGES (Cont'd)

Diagnostic	Message
FILE FUNCTION ILLEGAL	<p>User is working with direct access files and used one of the following options with the END, FILE or QUIT commands:</p> <ol style="list-style-type: none"> 1. The L or LOCAL option is not defined for direct access files. 2. The C or COPY option is not legal for direct access files not attached in WRITE mode.
FILE NAME CONFLICT	<p>User issued two or more XEDIT control card parameters which specified the same file name such as I=ABC and L=ABC (see appendix A on Batch Command Processing).</p>
FILE NOT XEDITABLE	<p>User has specified a file that does not contain a legal line but does contain something.</p>
<u>filename</u> CANNOT BE ACCESSED	<p>User has requested a file which is:</p> <ol style="list-style-type: none"> 1. An execute only or append only file. 2. Not local. 3. Not available from the user's permanent file area (for appropriate commands). Make sure your direct access file is attached in WRITE mode if you are editing a direct access file.
<u>filename</u> IS A LOCAL FILE	<p>User issued an L or LOCAL parameter on an END, FILE or QUIT command; XEDIT indicates that it has written any changes to the edit file as a NOS indirect access local file (see appendix D).</p>
<u>filename</u> REPLACED	<p>The edited file with any changes replaced the original NOS indirect access file under the same name as in the user's permanent file area (see appendix D).</p>
<u>filename</u> REWRITTEN	<p>The edited file with any changes has been copied back to the file filename. This message is mainly the result of editing direct access files attached in WRITE mode when the user issues an END, FILE or QUIT command with no parameters (see appendix D).</p>
<u>filename</u> SAVED	<p>The edited file was saved under NOS as a new indirect access permanent file (see appendix D).</p>

TABLE A-1. XEDIT DIAGNOSTICS AND MESSAGES (Cont'd)

Diagnostic	Message
FR COMMAND STACKING ERROR	XEDIT internal error. The FR XEDIT control card option has not been processed. Execution continues. Contact Customer Service.
ILLEGAL DELIMITER CHARACTER	XEDIT control card delimiter character is a space alphabetic numeric + - * / or up-arrow (↑).
ILLEGAL PARAMETER	<p>User issued a legal command, but specified one of the following:</p> <ol style="list-style-type: none"> 1. An alphabetic DELIMITT command delimiter. 2. Extra data after the last parameter. 3. A parameter on a command that does not require a parameter. 4. An RMARGIN value less than 10 or greater than 160. 5. Tab stops values not between 1 and 160 or tab stops value not in increasing order.
INPUT	User is in INPUT mode. User pressed the carriage return or entered the INPUT command and can now enter an indefinite number of lines that will be inserted immediately after the current pointer position.
LINE NUMBER NOT FOUND, COMMAND NOT EXECUTED	The line number prefix specified a line number which is not in the file. XEDIT does not execute the command and the pointer position remains the same.
LINE NUMBER TOO LARGE	A line number formed by the ALN, ALNS and RLN commands exceeded 99999.
LOCAL FILE ERROR	XEDIT internal error. Contact Customer Service.
NAME EDIT FILE	XEDIT wants the name of a file to edit. The user should enter the desired file name. If the file name is followed by a ",P" it will be read from the user's permanent file catalog if possible (that is, the file must be indirect access or must be direct access attachable in WRITE mode). If the file name is followed by a ",C" XEDIT will create a new local file under that name. See the XEDIT Batch Command Processing information in appendix C for a more detailed description of the C and P parameters.

TABLE A-1. XEDIT DIAGNOSTICS AND MESSAGES (Cont'd)

Diagnostic	Message
NO SUCH COMMAND	User issued an entry (in response to a double question mark) that was not a valid XEDIT command. Accordingly, either the entry itself is illegal or an improper separator appeared after the command.
NULL LINE - IN CREATION MODE	User issued a PRINT command, however, there is nothing for XEDIT to print since no lines have been created yet under CREATION mode. The user should issue any of the valid CREATION mode commands (see C parameter on the XEDIT control statement in appendix C).
STRING NOT FOUND	The specified string could not be located on the current line and the 0 option was used. The XEDIT pointer is not moved.
TRUNCATION AT <u>n</u>	A processed file line exceeds the maximum line width. XEDIT will truncate the <u>n</u> th line in the file (see Margin and Truncation Control).
XEDIT INTERNAL ERROR (ERD). NOTIFY CONSULTANT.	An XEDIT internal error has occurred; inform the central system's NOS Customer Service immediately.
XEDIT <u>version</u>	Begin text editing; XEDIT has successfully been called into execution. In in-line edit mode, this message is suppressed.
YOU DELETED THE ENTIRE FILE	XEDIT can not locate any line at which to position the pointer. XEDIT terminates execution; the original input file remains intact.
0 - IN CREATION MODE	User issued a WHERE command, however, the current line count is zero since no lines have been created yet under CREATION mode.

Most other terminal error messages not listed previously are issued by the NOS operating system. See appendix A of the CYBERNET Interactive Service Time-Sharing Usage Reference Manual listed in the Preface.

An on-line explanation of any of the previously listed messages can be obtained by entering the EXPLAIN command after the message is printed. See the EXPLAIN command for further details.

EDITING DIRECT ACCESS FILES

B

In addition to manipulating NOS indirect access files, XEDIT can apply to direct access files. While the command entry procedures and command functions are identical, XEDIT users differ when dealing with direct access in two ways:

- Direct access files are called differently (that is, with ATTACH commands). See page 2-3 for detailed information. Users should ATTACH their direct access files in WRITE mode so that they can be written on with the changes.
- When the user wants to replace the original direct access file with the edited changes, the XEDIT FILE or END commands should be used with no parameters. XEDIT knows when the user is using a direct access file. Through the use of the FILE and END commands, XEDIT will rewrite the user's file as requested. Figure B-1 illustrates this procedure.

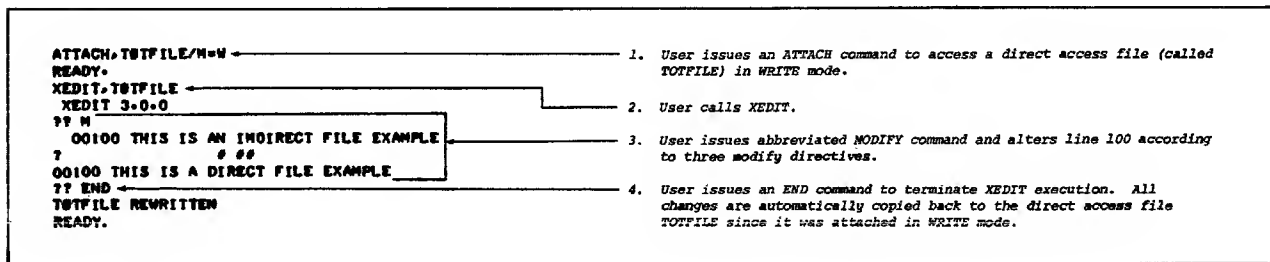


Figure B-1. Editing Direct Access Files

XEDIT BATCH COMMAND PROCESSING

C

In addition to normal interactive processing, XEDIT can also be executed in a batch environment, such as input via cards. This is accomplished by calling the XEDIT system with user-selected control card parameters.

To call the XEDIT system, in a batch environment, use either of the following control card formats:

← 80 column max →

XEDIT(fn,option1,...,optionn)dcs

or

XEDIT,fn,option1,...,optionn.dcs

where:

fn = the name of the file that the user wants to edit (see P option). This file is automatically rewound before and after processing, and if primary it is automatically sorted when necessary (new lines or text added). If this parameter is left off, the file to be edited defaults to the primary file, as in XEDIT,....

option₁ = B Forces normal batch processing (64 character set). This parameter is needed only if the user's NOS job using the XEDIT control statement is a time-sharing origin job (TXOT^{††}) such as in an NOS procedure file called interactively or when under the BATCH Subsystem. [†] If the user submits his NOS job from cards or with an NOS SUBMIT control statement, he does not use this parameter. This parameter (or Batch origin) causes any error in XEDIT to be fatal; XEDIT will immediately abort.

C Puts XEDIT into CREATION mode. This mode allows the user to create a new file from within XEDIT, without having to previously define the file with NOS control cards. The only legal initial XEDIT commands that are allowed are as follows:

<u>carriage return</u>	INSERT	TEOR
BRIEF	LISTAB	TOPNULL
DEFTAB	NOBELLS	TRIM
DELIMIT	RMARGIN	VERIFY
EXPLAIN	STOP	WHERE
HELP	TABS	Y
INPUT	TEOF	Z

Use of the C parameter will automatically generate a local file upon exit. If the file is to be primary, ^{††} either use the NEW control card prior to use or the PRIMARY control card after use.

[†]See the CYBERNET Interactive Service Time-Sharing Usage Reference Manual listed in the Preface.

^{††}See the NOS Reference Manual, Volume 1 listed in the Preface.

option_i = FR Allows the user to define a default set of commands which can be used to initialize the delimiter, tab character and tab stops, etc. Specifically, the FR parameter instructs XEDIT to scan the first line of the file being edited for the initial command (only one line of XEDIT input is allowed; for multiple commands use the Y or Z commands).

When using the FR option, XEDIT assumes that the first command to be executed will begin with the first nonblank character which occurs after at least two consecutive blanks and continues until the end of the line. This allows the use of both files with and without line numbers and the use of XEDIT commands in comment fields of the various compilers. Some examples follow:

```
COMPASS:      * Y/DEL;/DT$/TAB 10 20 30
FTNTS:        00100C Y/DEL;/DT$/TAB 10 20 30
MODIFY:       */ Y/DEL;/DT$/TAB 10 20 30
BASIC:        00100 REM Y/DEL;/DT$/TAB 10 20 30
```

The FR option is ignored when in CREATION mode. See C option.

I = lfn Causes XEDIT input commands and input directives to be read from the named local file lfn instead of the normal interactive prompts ?? and ? or the default NOS file INPUT. If the I = parameter is not used, XEDIT will take its input directives from the current INPUT[†] record of the NOS job deck (after an end-of-record mark, --EOR--, or a card multi-punched with a 7, 8, and 9 in column one). The file specified by an I = parameter is neither rewound before nor after XEDIT processing. Users should make sure that they include an XEDIT END or QUIT command as the last command in the current record of file lfn or XEDIT will abort.

If I = 0, XEDIT ignores the INPUT file and requires that all commands and command data must appear in the delimited command sequence dcs field on the XEDIT control card. Thus, if I = 0 is used, all lines of data used as input to commands such as INSERT must appear on the control card; otherwise XEDIT aborts.

CAUTION IN-LINE EDIT USERS

The use of the I= parameter when in in-line edit mode is illegal and will issue the message ERROR IN XEDIT ARGUMENTS (see appendix D).

L = lfn Causes normal XEDIT printed output to be placed on the named local file lfn instead of being printed at the interactive terminal. This file is neither rewound before nor after XEDIT processing. Also, XEDIT does not automatically shift the user's file by one character position to take care of erroneous printer carriage control characters.[†] Therefore, if the user wants to print the XEDIT output on a batch printer, it is advisable to add the following two NOS control statements after the XEDIT card:

```
REWIND(lfn)
COPYSBF(lfn)
```

If L = 0, no XEDIT output will be generated. The most common use of the L = 0 option would be when editing is desired as an intermediate control card or procedure file step in another user designed application.

NOTE

XEDIT inserts end-of-record (--EOR--) marks in the output file (lfn), if the user issues XEDIT control statements in succession which use the L= parameter.

[†]See the NOS Reference Manual, Volume 1 listed in the Preface.

- option_i = P Causes the indirect or direct access file fn that the user wants to edit to be retrieved from the user's permanent file catalog. Use of the P parameter eliminates the need for an NOS GET(lfn) or ATTACH(lfn/M=W) control statement. Indirect access files will not be made primary. Use the NOS[†] PRIMARY control card after exiting XEDIT, if a primary file is desired. The P parameter will only ATTACH direct access files in WRITE mode.
- option_i = NH Suppress the printing of the XEDIT header version message such as XEDIT 3.0.0. Note: when in in-line edit mode this parameter is ignored since the header is automatically suppressed.

dcs = a delimited command sequence; an optional list of commands separated by delimiters to be executed immediately by XEDIT before any other command input directives are processed such as the I = specified input file. The format of a delimited command sequence on the XEDIT control card is the same as a multiple command interactive entry (see the DELIMIT command) except that the initial delimiter character is specified by the first character of the delimited command sequence or by default (semi-colon). Thus, the form is:

char0 cmd1 char cmd2 char ... char cmdn

where:

char0 = same as char below but if not specified, it defaults to semi-colon(;).

char = any special character except a space alphanumeric + - * / or up-arrow (↑) which the user wants to establish as the delimit character. This character is used to separate the several commands or editing data that appear on the XEDIT control card. The default delimiter is semi-colon (;) and this delimiter remains in effect when normal editing continues unless a DELIMIT command is issued.

cmd = a legal XEDIT command, command data, or command directives which the user wants to process; XEDIT will execute each command according to its sequential (left to right) position in the list of component commands.

CAUTION IN-LINE EDIT USERS

The use of a delimited command sequence when in in-line edit mode will default the last command to END, will not prompt the user with a double question mark (??) and will give ERROR IN XEDIT ARGUMENTS if an I= parameter is specified (see appendix D).

CAUTION KEYPUNCH USERS

It is especially important that the correct octal display code representation of special characters be read by XEDIT when entering these characters via keypunch. Users should consult Tables B-2, B-6 and B-7 of the CYBERNET Time-Sharing Usage Reference Manual listed in the Preface. Note that the up-arrow (↑ octal 76) can not be represented directly with the Export Character Set (Table B-6) i.e., remote-batch. See the OCTCHANGE command instead.

[†]See the NOS Reference Manual, Volume 1 listed in the Preface.

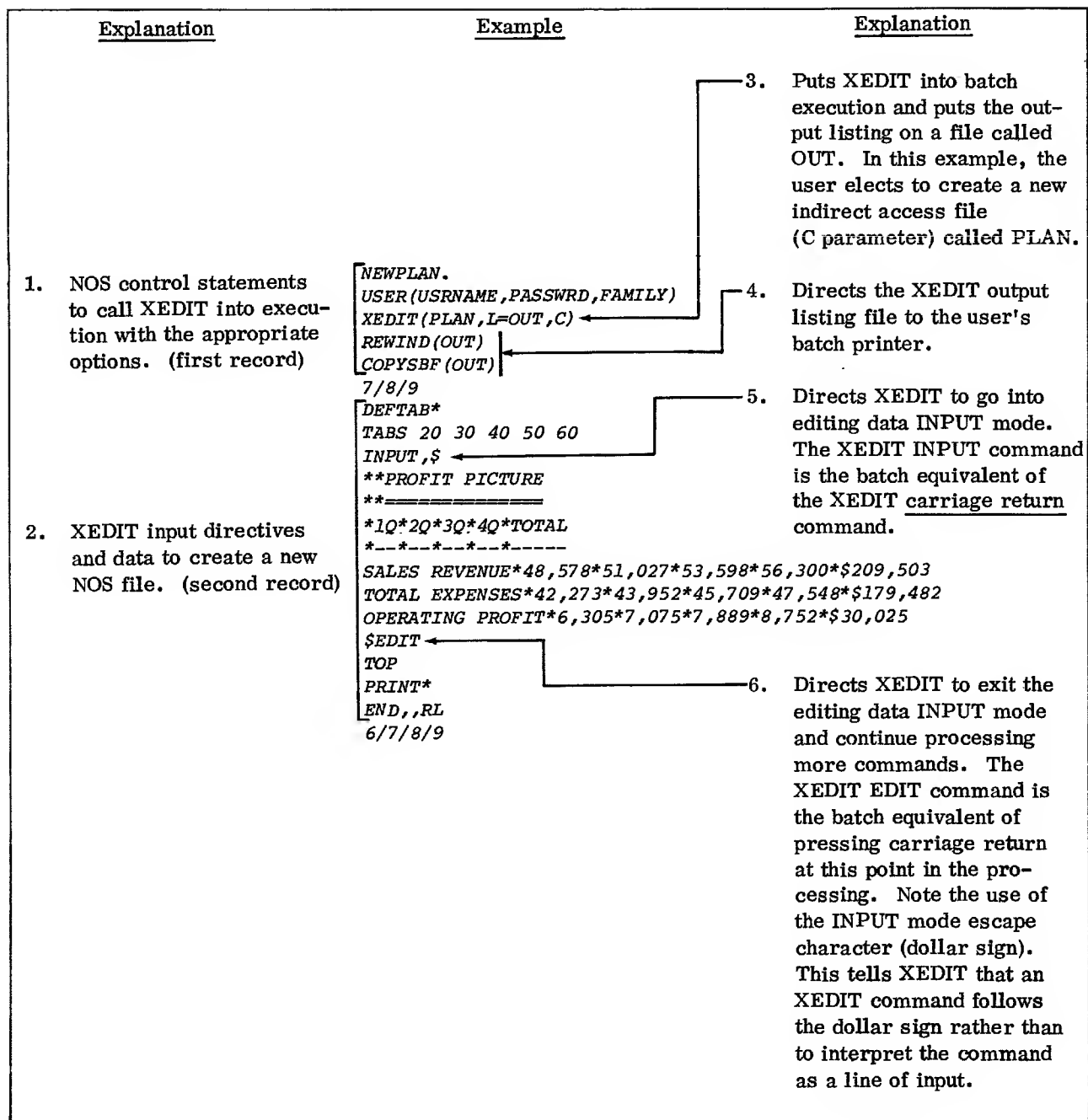


Figure C-1. Use of XEDIT Batch Processing Parameters to Create a New File

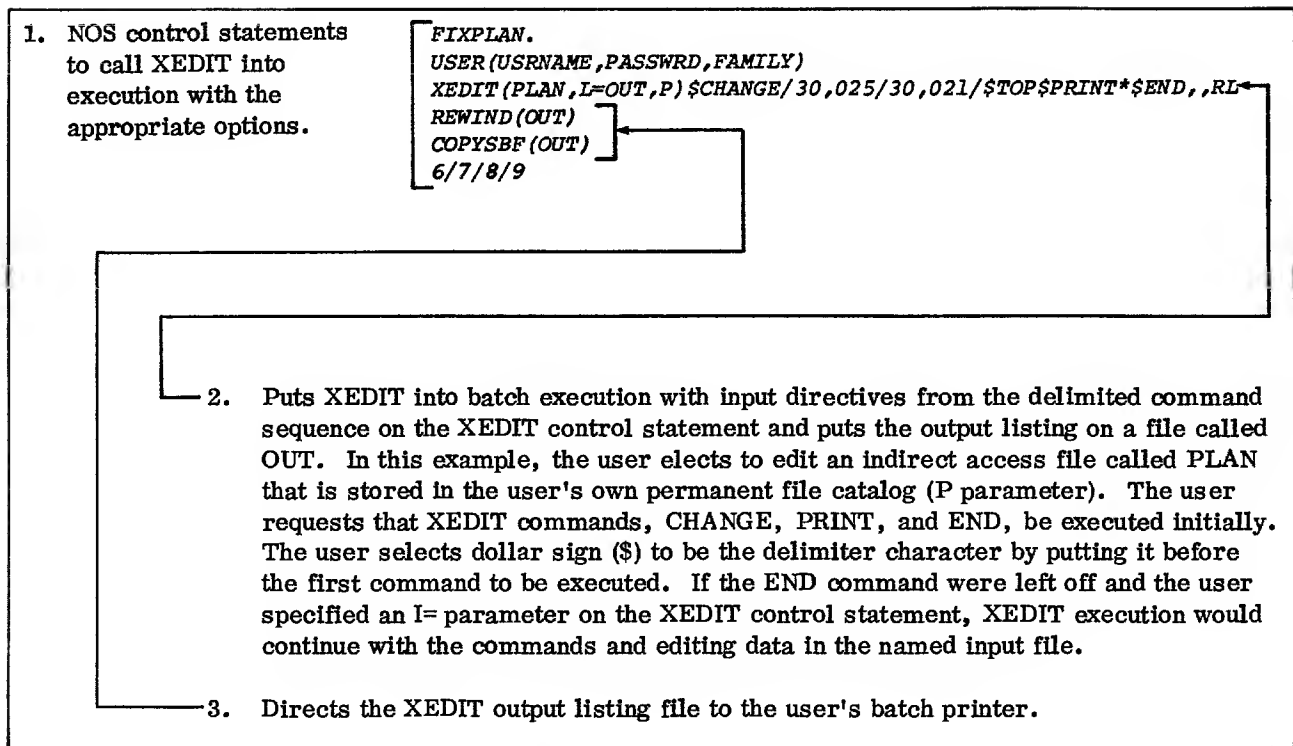


Figure C-2. Use of XEDIT Batch Processing Parameters to Edit a File

IN-LINE EDITOR USAGE

D

In addition to normal interactive processing, XEDIT can be used in in-line editing mode. This gives the user the ability to make quick "spot" changes to a file since the editor call and editing commands are entered all on the same line. Thus, waits for the prompt (??) are avoided and editing sessions can be completed in a shorter amount of time.

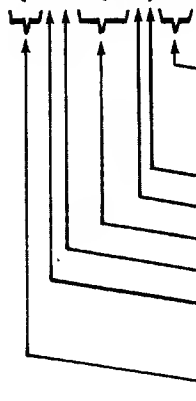
Some typical in-line editing calls are as follows:

Q.D3	This will delete the first three lines of the primary file (a file specified by OLD, NEW, LIB, or PRIMARY).
Q.P5	This will print the first five lines of the primary file.
Q.1000P5	This will print five lines of a line-numbered primary file, beginning at line 1000.
Q.C/ABC/XYZ/	This will locate string ABC and change it to XYZ in the primary file.
Q.500C/ABC/XYZ/0	This will change string ABC to string XYZ, but only if the string is found on line 500.
Q.I2 ? CAT ? DOG	This will insert strings CAT and DOG after the first line of the primary file.

NOTE

The editor prompts the user with (?).

Q.N3;I ? MOUSE	This will insert string MOUSE after the third line of the primary file.
Q.+XYQM*;;##	This tells XEDIT to delete the first three characters of all lines in the primary file and do not verify the changes.



Explanation:

YQM command input directives (# means delete one character) are on the same line as the command because the plus (+) prefix character was used.

Default delimiter is semicolon (;).

Specifies that command applies to all lines.

See the YQMOD command description.

Tells XEDIT to suppress change verification printing.

Tells XEDIT to look for the input directives on the same line as this statement immediately following the command specification.

Calls XEDIT into in-line editing mode. The command, XEDIT, can be used also, but prompts the user with (??) when all of the in-line commands have been executed (see appendix C).

Q,REPORT.2000P This calls XEDIT to print line 2000 of direct access (see NOS[†] ATTACH or DEFINE commands) or secondary indirect access (see NOS GET command) file REPORT.

Q.\$CS/MONTH;THUS/MONTH;HOWEVER/\$P2
This changes the in-line editing command delimiter character from semicolon (;) to dollar sign (\$) so that a change string by context can be made with a string which contains semicolon (;). Two lines are then printed.

From the above examples, the in-line editing format should be clear; however, the following presents a formal definition of the in-line editor.

To edit a primary file in in-line edit mode use the following syntax:

←80 max→

Q.dcs

or

←80 column max→

Q,,option1,...,optionn.dcs

To edit a direct access or secondary indirect access file in in-line edit mode, use the following syntax:

←80 column max→

Q,fn,option1,...,optionn.dcs

where:

fn = same as appendix C.

option_i = same as appendix C, except under in-line edit mode the I= parameter is illegal and will produce the message ERROR IN XEDIT ARGUMENTS.

dcs = same as appendix C, except the last command always defaults to END unless an END, E, QUIT or Q command is explicitly specified. See the additional rules specified below.

The following basic rules apply only to in-line edit mode (NOS Q commands):

- The entry header message such as XEDIT 3.0.0 will not print.
- The editor will always return the user back to NOS control after the last command has been processed. The last command will default to END if an END, E, QUIT, or Q command is not explicitly specified.

[†] See the NOS Reference Manual, Volume 1 listed in the Preface.

- If an error occurs any place within the command string (dcs), the editor will exit in-line edit mode giving only the normal error messages and will abort. For the user's protection if an abort occurs, no changes against the edited file will be kept (even if the user is editing a direct access file). The message ABORTED will not appear (see appendix A).
- If the user is editing an indirect access file, the message filename IS A LOCAL FILE will not be issued upon completion of the last command in the list unless the user has explicitly used the L parameter, such as END,,L (see pages 2-67 through 2-69 and appendix A).
- If the user is editing a direct access file, the appropriate message such as filename REWRITTEN will always be issued (see pages 2-67 through 2-69 and appendix A).
- If the user is editing an indirect access file and explicitly specifies a parameter on END, E, QUIT, or Q such as S, R, or L, the appropriate message such as filename SAVED will be issued (see pages 2-67 through 2-69 and appendix A).
- If no command list (dcs) is specified, the message ERROR IN XEDIT ARGUMENTS is issued and processing returns to NOS.

INDEX

A command 1-6, 2-16
A postfix character 2-27
Aborted XEDIT execution A-1
 Also see "STOP command"
ADD command 1-6, 2-5, 2-16
Adding end-of-record marks 2-43
Adding line numbers 2-38
Adding lines 2-29 to 2-32
Adding strings 2-16
ADDLN command 1-7, 2-39
ADDLNS command 1-7, 2-39
Advancing the file pointer 2-13
ALN command 1-7, 2-39
ALNS command 1-7, 2-39
Altering a permanent file 1-3, 2-66
Anchor (def.) 2-27
And sign (&) modify directive 2-22
ASCII character code 1-1
ATTACH command 1-5, 2-3

B command 1-5, 2-14
B parameter C-1
Bad lines
 Deleting 2-42
 Locating 2-15
Batch command processing C-1
Beginning-of-file position 2-14
BOTTOM command 1-5, 2-14
Bottom-of-file position 2-14
BR command 1-8, 2-6
Break key 1-5
BRIEF command 1-8, 2-6
BRIEF mode 2-6

C command 1-6, 2-17
C mode 1-10
C parameter C-1
Calling the in-line editor Q D-1
Calling XEDIT 1-5, 2-1 to 2-4
Canceling editing operations 2-53
Caret prefix 2-5
Carriage return command 2-34
CHANGE command 1-6, 2-17
CHANGES command 1-6, 2-17
Changing string contents 2-17
 Also see "Modifying strings"

Circular line search 1-5, 2-10
Column numbers 2-4, 2-23
Commands (XEDIT)
 A 1-6, 2-16
 ADD 1-6, 2-5, 2-16
 ADDLN 1-7, 2-39
 ADDLNS 1-7, 2-39
 ALN 1-7, 2-39
 ALNS 1-7, 2-39
 ATTACH 1-5, 2-3
 B 1-5, 2-14
 BOTTOM 1-5, 2-14
 BR 1-8, 2-6
 BRIEF 1-8, 2-6
 C 1-6, 2-17
 Carriage return 2-34
 CHANGE 1-6, 2-17
 CHANGES 1-6, 2-17
 COPY 1-8, 2-48
 COPYD 1-8, 2-49
 Ⓒ 2-34
 D 1-7, 2-29
 DASH (-) 2-55
 DBADL 1-7, 2-42
 DBL 1-7, 2-42
 DEFTAB 1-9, 2-60
 DEL 1-9, 2-56
 DELETE 1-7, 2-29
 DELETELN 1-7, 2-40
 DELIMIT 1-9, 2-56
 DEOF 1-7, 2-43
 DEOR 1-8, 2-42
 DF 1-7, 2-43
 DLB 1-8, 2-44
 DLBLANKS 1-8, 2-44
 DLN 1-7, 2-40
 DR 1-8, 2-42
 E 1-10, 2-67
 EDIT 2-36
 eEDIT 1-7, 2-36
 END 1-10, 2-67
 EXPLAIN 1-8, 2-52
 F 1-10, 2-7, 2-65
 FBADL 1-5, 2-15
 FBL 1-5, 2-15

Commands (XEDIT) (Contd)

FILE 1-10, 2-7, 2-64
 File manipulation 2-47 to 2-51
 Find line number 1-5, 2-9
 FINDLL 1-9, 2-65
 FLL 1-9, 2-65
 General 1-8
 Generalized 2-51
 GET 1-5, 2-3
 H 1-8, 2-52
 HELP 1-4, 2-52
 I 1-7, 2-5, 2-33
 IB 1-7, 2-5, 2-34
 In-line editing D-1
 INPUTe 1-7, 2-36
 INSERT 1-7, 2-5, 2-33
 INSERTB 1-7, 2-5, 2-34
 Issuing 2-4
 L 1-5, 1-10, 2-11
 Line editing 2-29
 Line number 2-38
 LISTAB 1-9, 2-61
 ln 1-5, 2-10
 LOCATE 1-5, 2-11
 LT 1-9, 2-61
 M 1-6, 2-21
 Margin control 2-62
 Minus (-) 2-55
 Miscellaneous editing 1-7, 2-41
 MODIFY 1-6, 2-5, 2-21
 Multiple 1-9
 N 1-5, 2-13
 NB 1-8, 2-52
 NEW 1-5, 2-2
 NEXT 1-5, 2-13
 NOBELLS 1-8, 2-52
 OC 1-8, 2-45
 OCTCHANGE 1-8, 2-45
 OLD 1-5, 2-2
 P 1-6, 2-15
 Parameters 1-3, 2-8
 Period (.) 2-55
 Pointer movement 2-10 to 2-16
 Prefixes 1-5, 2-7, 2-9
 PRINT 1-6, 2-15
 Q 1-10, 2-67
 Q in-line edit NOS D-1
 QM 1-6, 2-23
 QMOD 1-6, 2-5, 2-23
 QUIT 1-10, 2-67
 R 1-7, 2-5, 2-32
 READ 1-8, 2-51

Commands (XEDIT) (Contd)

READP 1-8, 2-50
 Repeating 2-55
 REPLACE 1-7, 2-5, 2-32
 REPLACELN 1-7, 2-41
 REST 1-9, 2-53
 RESTORE 1-9, 2-53
 RLN 1-7, 2-41
 RM 1-9, 2-62
 RMARGIN 1-9, 2-62
 STOP 1-10, 2-66
 String editing 2-16 to 2-24
 String search control 2-25 to 2-28
 Summary list 1-5 to 1-7
 Syntax rules 1-3, 2-4
 T 1-6, 2-14
 TAB 1-9, 2-61
 Tab control 1-9, 2-60
 TABS 1-9, 2-61
 TEOF 2-54
 TEOR 2-54
 TN 1-7, 2-37
 TOP 1-6, 2-14
 TOPNULL 1-7, 2-37
 Trailing blank control 2-25
 TRIM 1-6, 2-25
 TRUNC 1-9, 2-63
 TRUNCATE 1-9, 2-63
 Truncation control 2-62
 V 1-9, 2-6
 VERIFY 1-9, 2-6
 Verifying results 1-3, 2-6
 W 1-9, 2-54
 WEOF 1-8, 2-44
 WEOR 1-8, 2-43
 WF 1-8, 2-44
 WHERE 1-9, 2-54
 Window control 2-26 to 2-28
 WM 1-6, 2-26
 WMARGIN 1-6, 2-26
 WR 1-8, 2-43
 Y 1-9, 2-7, 2-57
 YQM 1-6, 2-5, 2-24
 YQMOD 1-6, 2-5, 2-24
 Z 1-9, 2-7, 2-57
 Conventions (XEDIT) 1-3, 2-4 to 2-8
 Summary list 1-3
 Converting octal strings 2-45
 COPY command 1-8, 2-48
 Copy mode 1-10
 COPYD command 1-8, 2-49
 CR command 2-34, 2-35
 Creation mode C-1

- D command 1-7, 2-29
- Dash (-) command 2-55
- DBADL command 1-7, 2-42
- DBL command 1-7, 2-42
- Defining a window 2-26
- DEFTAB command 1-9, 2-60
- DEL command 1-9, 2-56
- DELETE command 1-7, 2-29
- DELETELN command 1-7, 2-40
- Deleting bad lines 2-42
- Deleting blanks 2-44
- Deleting end-of-record marks 2-43
- Deleting line numbers 2-40
- Deleting lines 2-29
- Deleting strings 2-17, 2-19
- DELIMIT command 1-9, 2-56
- Delimited command sequence 2-56, C-3
- Delimiters
 - Command delimiters 2-56
 - Diagnostic A-1
 - String delimiters 2-3, 2-6
- DEOF command 1-7, 2-43
- DEOR command 1-8, 2-42
- DF command 1-7, 2-43
- Diagnostics (XEDIT) A-1 to A-3
- Direct access files 2-4, B-1
- Display code
 - See "Octal strings" 2-45
- DLB command 1-8, 2-44
- DLBLANKS command 1-8, 2-44
- DLN command 1-7, 2-40
- DR command 1-8, 2-42
- DT command 1-9, 2-60

- E command 1-10, 2-67
- EDIT command 2-36
- EDIT mode 2-36
- Editing
 - Data 1-3, 2-5
 - Also see "Input mode"
 - Line numbers 2-38 to 2-41
 - Lines 2-29 to 2-37
 - Miscellaneous editing 2-42
 - Octal strings 2-45
 - Strings 2-16 to 2-24
- eEDIT command 1-7, 2-36
- END command 1-10, 2-67
- End of file message 2-8
- End-of-file marks
 - Deleting marks 2-43
 - Inserting marks 2-43

- End-of-file positioning
 - See "End of information positioning" 2-9
- End-of-information positioning 2-9
- End-of-record marks 1-1, 1-2, A-1
 - Deleting marks 2-43
 - Inserting marks 2-43
- Entering editing data 1-3, 2-5
- Executing XEDIT 1-5, 2-1 to 2-4
 - Termination 1-10, 2-65 to 2-69
- EXPLAIN command 1-8, 2-52

- F command 1-10, 2-7, 2-65
- FBADL command 1-5, 2-15
- FBL command 1-5, 2-15
- FILE command 1-10, 2-7, 2-64
- File commands 1-8
- Files
 - Direct Access 2-4, B-1
 - File names A-1, A-2
 - File pointer 1-3, 2-8 to 2-16
 - Indirect access 1-1, 2-1 to 2-3
 - Manipulation by command 2-47 to 2-51
 - Primary 2-1 to 2-3
 - Saving the edited file 1-10, B-1
- Find line number command 1-5, 2-9
- FINDLL command 1-9, 2-65
- FLL command 1-9, 2-65
- FR parameter C-2

- Generalized commands 2-51
- GET command 1-5, 2-3

- H command 1-8, 2-52
- HELP command 1-4, 2-52

- I command 1-7, 2-5, 2-33
- I= parameter C-2
- IB command 1-7, 2-5, 2-34
- In-line editor D-1
- Indirect access files 1-1, 2-1 to 2-3
- INPUT command 1-7, 2-5, 2-33
- INPUT mode 2-34
- INPUTe command 1-7, 2-36
- INSERT command 1-7, 2-5, 2-33
- INSERTB command 1-7, 2-5, 2-34
- Inserting blank lines 2-37
- Inserting end-of-record marks 2-43
- Inserting line numbers 2-38
- Inserting lines 2-29 to 2-32
- Inserting strings 2-16, 2-20

Internal display codes 2-46
 Also see "Octal strings"
 Interrupting XEDIT printing 1-5
 Interrupting XEDIT processing 1-3, 2-7
 Issuing XEDIT commands 1-3

Keypunch users C-3

L command 1-5, 2-11
 L mode 1-10
 L= parameter C-2
 Line count 2-54
 Line editing 2-29 to 2-37
 Line number editing commands 2-38 to 2-41
 Line number search
 See "Find line number command"
 Line size 1-3
 Line truncation 2-3
 Line width
 See "Margin and truncation control"
 LISTAB command 1-9, 2-61
 Listing file lines 2-15
 In command 1-5, 2-10
 LOCAL mode 1-10
 LOCATE command 1-5, 2-11
 Locating bad lines 2-15
 Locating lines by line number 2-10
 Locating lines by string 2-10
 LT command 1-9, 2-61

M command 1-6, 2-21
 Margin and truncation control 2-62
 Maximum line width
 See "Margin and truncation control"
 Merging files 2-50
 Messages (XEDIT) A-1 to A-3
 See also "EXPLAIN command"
 Minus (-) command 2-55
 Miscellaneous editing commands 1-7, 2-41
 MODIFY command 1-6, 2-5, 2-21
 MODIFY directives 2-22
 Modifying strings 2-21, 2-23
 Also see "Changing string contents"
 Multi-record files 1-1, 1-2
 Multiple commands 1-9
 Multiple entries in a single line 2-55 to 2-59

N command 1-5, 2-13
 NB command 1-8, 2-52
 NEW command 1-5, 2-2
 New files 2-2

NEXT command 1-5, 2-13
 NH parameter C-3
 NOBELLS command 1-8, 2-52
 Number sign (#) MODIFY directive 2-22

OC command 1-8, 2-45
 Octal strings 2-45
 OCTCHANGE command 1-8, 2-45
 OLD command 1-5, 2-2
 Old files 2-2

P command 1-6, 2-15
 P parameter C-3
 Period (.) command 2-55
 Plus (+) postfix 2-6, 2-25, 2-54
 Plus (+) prefix 2-4, 2-59
 Pointer movement 1-5, 2-8 to 2-16
 Commands 2-9 to 2-16
 Conventions 2-8, 2-9
 Postfix characters 1-5
 Pound sign (#) MODIFY directive 2-22
 Prefixes (command) 1-5, 2-7, 2-9
 Primary files 2-1 to 2-3
 PRINT command 1-6, 2-15

Q command 1-10, 2-67
 Q in-line edit NOS command D-1
 QM command 1-6, 2-33
 QMOD command 1-6, 2-5, 2-23
 QUIT command 1-10, 2-67

R command 1-7, 2-5, 2-32
 R mode 1-10
 READ command 1-8, 2-51
 READP command 1-8, 2-50
 Repeating command execution 2-55
 REPLACE command 1-7, 2-5, 2-32
 REPLACE mode 1-10
 REPLACELN command 1-7, 2-41
 Replacing line numbers 2-41
 Replacing lines 2-32
 Replacing permanent files
 See "Saving the edited file"
 Replacing strings 2-17
 Repositioning pointer 2-14
 REST command 1-9, 2-53
 RESTORE command 1-9, 2-53
 Restoring edited file contents 2-53
 Reversing pointer movement 2-13
 RLN command 1-7, 2-41
 RM command 1-9, 2-62
 RMARGIN command 1-9, 2-62

- S mode 1-10
- SAVE mode 1-10
- Saving the edited file 1-10, B-1
- Secondary files 2-3
- Selective command samples 1-4
- Slash (/) prefix 1-5, 2-9
- STOP command 1-10, 2-66
- String editing 2-16 to 2-24
- String-line (def.) 2-8
- Strings (def.) 2-6
 - Delimiters 1-3, 2-6
 - Search controls 1-6, 2-25
- Suspending XEDIT 2-64
- Syntax (command) 2-4
- T command 1-6, 2-14
- TAB command 1-9, 2-61
- Tab control commands 1-9, 2-60
- TABS command 1-9, 2-61
- TEOF command 2-54
- TEOR command 2-54
- Terminating XEDIT execution 1-10, 2-65 to 2-67
- TN command 1-7, 2-37
- TOP command 1-6, 2-14
- TOPNULL command 1-7, 2-37
- Trailing blanks 2-25
- TRIM command 1-6, 2-25
- TRIM mode 2-25
- TRUNC command 1-9, 2-63
- TRUNCATE command 1-9, 2-63
- Truncation (line) 2-62
 - See "Margin and truncation control"
- Up arrow (↑) prefix 1-5
- Up arrow MODIFY directive 2-22
- Using the window 2-27
- V command 1-9, 2-6
- VERIFY command 1-9, 2-6
- VERIFY mode 1-9, 2-6
- Verifying XEDIT operations 1-3

- W command 1-9, 2-54
- W postfix character 2-27
- WEOF command 1-8, 2-44
- WEOR command 1-8, 2-43
- WF command 1-8, 2-44
- WHERE command 1-9, 2-54
- Window definition and usage 2-26 to 2-28
- WM command 1-6, 2-26
- WMARGIN command 1-6, 2-26
- WR command 1-8, 2-43
- X prefix 1-5, 2-7
- XEDIT (def.) 1-1
 - Benefits 1-1
 - Command parameters 2-8
 - Command summary 1-5 to 1-10
 - Control card parameters C-1 to C-3
 - Convention summary 1-2, 2-4 to 2-8
 - Diagnostics A-1 to A-3
 - Execution of 2-1 to 2-4
 - Features 1-1
 - File manipulation 2-47 to 2-51
 - Functions 1-1
 - In-line editing mode D-1
 - Line editing 2-29 to 2-37
 - Line number editing 2-38 to 2-41
 - Miscellaneous editing 2-41
 - Multiple entries 2-55 to 2-59
 - NOS command format C-1
 - Sample user sessions 1-4, 1-8 to 1-15, B-1
 - Terminating execution 2-64 to 2-69
- Y command 1-9, 2-7, 2-57
 - Repeating Y command execution 2-55
- YQM command 1-6, 2-5, 2-24
- YQMOD command 1-6, 2-5, 2-24
- Z command 1-9, 2-7, 2-57
 - Repeating Z command execution 2-55
- .n command 1-9, 2-55
- n command 1-9, 2-55

COMMENT SHEET

MANUAL TITLE XEDIT User Information Manual

PUBLICATION NO. 76071000 REVISION C November 1977

FROM:

NAME: _____

BUSINESS
ADDRESS: _____

COMMENTS:

This form is not intended to be used as an order blank. Your evaluation of this manual will be welcomed by Control Data Corporation. Any errors, suggested additions or deletions, or general comments may be made below. Please include page number references.

CUT ALONG LINE

STAPLE

STAPLE

FOLD

FOLD

FIRST CLASS
PERMIT NO. 8241

MINNEAPOLIS, MINN.

BUSINESS REPLY MAIL

NO POSTAGE STAMP NECESSARY IF MAILED IN U.S.A.

POSTAGE WILL BE PAID BY

CONTROL DATA CORPORATION

Publications and Graphics Division

P. O. BOX 0 HQC02C

MINNEAPOLIS, MINNESOTA 55440

ATTN: DATA SERVICES PUBLICATIONS

CUT ALONG LINE

FOLD

FOLD

IMPORTANT REGULATORY NOTICE

Users of Control Data services should be aware that the rules and regulations of the United States and International Telecommunications Regulatory Agencies prohibit Control Data from using communications services it leases from domestic, international and foreign communications carriers to transmit information for its users which is not part of a "single integrated" data processing service. All information transmitted must be directly related to the data processing applications or service provided by Control Data and unprocessed information shall not be allowed through the service between user terminals, either directly or on a store and forward basis. Noncompliance with these rules and regulations may force Control Data to discontinue the users' data processing service.

CORPORATE HEADQUARTERS
8100 34TH AVENUE SOUTH
MINNEAPOLIS, MINNESOTA
MAILING ADDRESS • BOX 0, MPLS., MINN. 55440
SALES OFFICES AND SERVICE CENTERS
IN MAJOR CITIES THROUGHOUT THE WORLD

